
ARMA Nets: Expanding Receptive Field for Dense Prediction

Jiahao Su¹ Shiqi Wang² Furong Huang¹
jiahaosu@umd.edu 161170041@smail.nju.edu.cn furongh@cs.umd.edu
¹University of Maryland, College Park, MD USA ²Nanjing University, Nanjing, China

Abstract

Global information is essential for dense prediction problems, whose goal is to compute a discrete or continuous label for each pixel in the images. Traditional convolutional layers in neural networks, initially designed for image classification, are restrictive in these problems since the filter size limits their receptive fields. In this work, we propose to replace any traditional convolutional layer with an autoregressive moving-average (ARMA) layer, a novel module with an adjustable receptive field controlled by the learnable autoregressive coefficients. Compared with traditional convolutional layers, our ARMA layer enables explicit interconnections of the output neurons, and learns its receptive field by adapting the autoregressive coefficients of the interconnections. ARMA layer is adjustable to different types of tasks: for tasks where global information is crucial, it is capable of learning relatively large autoregressive coefficients to allow for an output neuron’s receptive field covering the entire input; for tasks where only local information is required, It can learn small or near zero autoregressive coefficients and automatically reduces to a traditional convolutional layer. We show both theoretically and empirically that the effective receptive field of networks with ARMA layers (named as ARMA networks) expands with larger autoregressive coefficients. We also provably solve the instability problem of learning and prediction in the ARMA layer through a re-parameterization mechanism. Additionally, we demonstrate that ARMA networks substantially improve their baselines on challenging dense prediction tasks including video prediction and semantic segmentation.

1 Introduction

Convolutional layers in neural networks have many successful applications for machine learning tasks. Each output neuron encodes an input region of the network measured by the *effective receptive field* (ERF) [24]. A large ERF that allows for sufficient global information is needed to make accurate predictions; however, a simple stack of convolutional layers does not effectively expand ERF. Convolutional neural networks (CNNs) typically encode global information by adding downsampling (pooling) layers, which coarsely aggregate global information. A fully-connected classification layer subsequently reduces the entire feature map to an output label. Downsampling and fully-connected layers are suitable for image classification tasks where only a single prediction is needed. But they are less effective, due to potential loss of information, in dense prediction tasks such as semantic segmentation and video prediction, where each pixel requests a prediction. Therefore, it is crucial to introduce mechanisms that enlarge ERF without too much information loss.

Naive approaches to expanding ERF, such as deepening the network or enlarging the filter size, drastically increase the model complexity, which results in expensive computation, difficulty in optimization, and susceptibility to overfitting. Recently advanced architectures have been proposed to expand ERF, including encoder-decoder structured networks [29], dilated convolutional networks [39, 40], and non-local attention networks [33]. However, encoder-decoder structured networks could lose high-frequency information due to the downsampling layers. Dilated convolutional networks could

suffer from the gridding effect while the ERF expansion is limited, and non-local attention networks are expensive in training and inference.

We introduce a novel *autoregressive-moving-average* (ARMA) layer that enables adaptive receptive field by explicit interconnections among its output neurons. Our ARMA layer realizes these interconnections via extra convolutions on output neurons, on top of the convolutions on input neurons as in a traditional convolutional layer. We provably show that an ARMA network can have arbitrarily large ERF, thus encoding global information, with minimal extra parameters at each layer. Consequently, an ARMA network can flexibly enlarge its ERF to leverage global knowledge for dense prediction without reducing spatial resolution. Moreover, the ARMA networks are independent of the architectures above including encoder-decoder structured networks, dilated convolutional networks and non-local attention networks.

A significant challenge in ARMA networks lies in the complex computations needed in both forward and backward propagations — simple convolution operations are not applicable since the output neurons are influenced by their neighbors and thus interrelated. Another challenge in ARMA networks is instability — the additional interconnections among the output neurons could recursively amplify the outputs and lead them to infinity. We address both challenges in this paper.

Summary of Contributions

- We introduce a novel ARMA layer that is a plug-and-play module substituting convolution layers in neural networks to allow flexible tuning of their ERF, adapting to the task requirements and improving performance in dense prediction problems.
- We recognize and address the problems of *computation* and *instability* in ARMA layers. **(1)** To reduce computational complexity, we develop FFT-based algorithms for both forward and backward passes; **(2)** To guarantee stable learning and prediction, we propose a *separable ARMA layer* and a re-parameterization mechanism that ensures the layer to operate in a stable region.
- We successfully apply ARMA layers in ConvLSTM network [38] for pixel-level multi-frame video prediction and U-Net model [29] for medical image segmentation. ARMA networks substantially outperform the corresponding baselines on both tasks, suggesting that our proposed ARMA layer is a general and useful building block for dense prediction problems.

2 Related Works

Dilated convolution [14] enlarges the receptive field by upsampling the filter coefficients with zeros. Unlike encoder-decoder structure, dilated convolution preserves the spatial resolution and is thus widely used in dense prediction problems, including semantic segmentation [6, 23, 39], and objection detection [9, 19]. However, dilated convolution by itself creates gridding artifacts if its input contains higher frequency than the upsampling rate [40], and the inconsistency of local information hampers the performance of the dilated convolutional networks [34]. Such artifacts can be alleviated by extra anti-aliasing layer [40], group interacting layer [34] or spatial pyramid pooling [7].

Deformable convolution allows the filter shape (i.e. locations of the incoming pixels) to be learnable [10, 15, 41]. While deformable convolution focuses on adjusting the filter *shape*, our ARMA layer aims to expand the filter *size* adaptively.

Non-local attention network [33] inserts non-local attention blocks between the convolutional layers. A non-local attention block computes a weighted sum of all input neurons for each output neuron, similar to attention mechanism [32]. In practice, non-local attention blocks are computationally expensive, thus they are typically inserted in the upper part of the network (with lower resolution). In contrast, our ARMA layers are economical (see section 4), and can be used throughout the network.

Encoder-decoder structured network pairs each downsampling layer with another upsampling layer to maintain the resolution, and introduces skip-connection between the pair to preserve the high-frequency information [23, 29]. Since the shortcut bypasses the downsampling/upsampling layers, the network has a small receptive field for the high-frequency components. A potential solution is to augment upsampling with non-local attention block [26] or ARMA layer (section 6).

Spatial recurrent neural networks apply recurrent propagations over the spatial domain [4, 16, 22, 27, 31], and learns the affinity between neighboring pixels [21]. Most of these prior works consider nonlinear recurrent neural networks, where the activation between recursions prohibits an efficient

FFT-based algorithm. In contrast, our proposed ARMA layer is equivalent to a linear recurrent neural network, where the spatial recurrences in ARMA layer can be efficiently evaluated using FFT.

3 ARMA Neural Networks

In this section, we introduce a novel *autoregressive-moving-average* (ARMA) layer, and analyze its ability to expand *Effective Receptive Field* (ERF) in neural networks. The analysis is further verified by visualizing the ERF with varying network depth and strength of autoregressive coefficients.

3.1 ARMA Layer

A traditional convolutional layer is essentially a *moving-average* model [3], $\mathcal{Y}_{:,t} = \sum_{s=1}^S \mathcal{W}_{:,t,s} * \mathcal{X}_{:,s}$, where the *moving-average coefficients* $\mathcal{W} \in \mathbb{R}^{K_m \times K_m \times T \times S}$ is parameterized by a 4th-order kernel (K_m is the filter size, and S, T are input/output channels), $:$ denotes all elements from the specified coordinate, and $*$ denotes convolution between an input feature and a filter.

As motivated in the introduction, we introduce a novel ARMA layer, that enables adaptive receptive field by introducing explicit interconnections among its output neurons as illustrated in Figure 1. Our ARMA layer realizes these interconnections by introducing extra convolutions on the outputs, in addition to the convolutions on the inputs as in a traditional convolutional layer. As a result, in an ARMA layer, each output neuron can be affected by an input pixel faraway through interconnections among the output neurons, thus receives global information. Formally, we define ARMA layer in Definition 1.

Definition 1 (ARMA layer). An ARMA layer is parameterized by a moving-average kernel (coefficients) $\mathcal{W} \in \mathbb{R}^{K_m \times K_m \times S \times T}$ and an autoregressive kernel (coefficients) $\mathcal{A} \in \mathbb{R}^{K_a \times K_a \times T}$. It receives an input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and returns an output $\mathcal{Y} \in \mathbb{R}^{I'_1 \times I'_2 \times T}$ with an ARMA model:

$$\mathcal{A}_{:,t} * \mathcal{Y}_{:,t} = \sum_{s=1}^S \mathcal{W}_{:,t,s} * \mathcal{X}_{:,s} \quad (1)$$

Remarks: (1) Since the output interconnections are realized by convolutions, the ARMA layer maintains the *shift-invariant* property. (2) The ARMA layer *reduces* to a traditional layer if the autoregressive kernel \mathcal{A} represents an identical mapping. (3) The ARMA layer is a *plug-and-play* module that can replace *any* convolutional layer, adding $K_a^2 T$ extra parameters negligible compared to $K_w^2 ST$ parameters in a traditional convolution layer. (4) Different from traditional layer, computing Equation 1 and its backpropagation is nontrivial, studied in section 4.

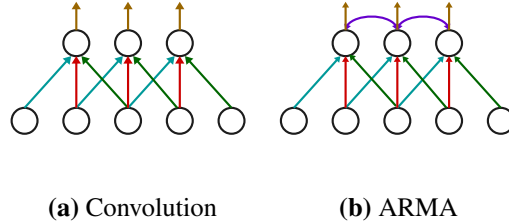


Figure 1: The ARMA layer introduces interconnections among output neurons explicitly.

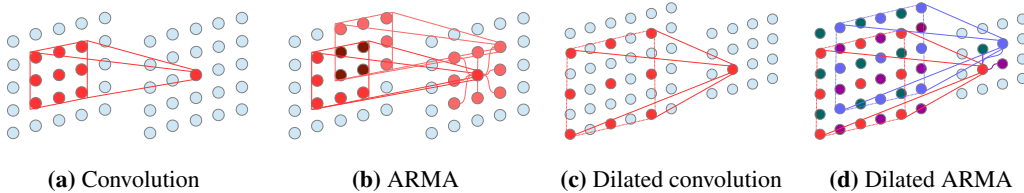


Figure 2: Diagrams of receptive field. In ARMA layer (b), each output neuron receives its neighbors' receptive field. In (d), ARMA's autoregression fills the gaps created by dilated convolution.

Our ARMA layer is complementary to dilated convolutional layer, deformable convolutional layer, non-local attention block and encoder-decoder architecture, and can be combined with each of them. For instance, *dilated ARMA layer*, illustrated in Figure 2d, removes the gridding effect caused by dilated convolution — the autoregressive kernel can be interpreted as an anti-aliasing filter.

The motivation of introducing ARMA layer is to enlarge the effective input region for each network output without increasing the filter size or network depth, thus avoiding the difficulties in training

larger or deeper models. As illustrated in Figure 2, each output neuron in a traditional convolutional layer (Figure 2a) only receives information from a small input region (the filter size). However, an ARMA layer enlarges the region from a local small one to a larger one (Figure 2b), and enables an output neuron to receive information from a faraway input neuron through the connections to its neighbors. Now we formally introduce the concept of *effective receptive field* (ERF) to characterize the effective input region. And we will provably show that an ARMA network can have arbitrarily large ERF with a single extra parameter at each layer in Theorem 3 in the following subsection.

3.2 Effective Receptive Field

Effective receptive field (ERF) [24] measures the area of the input region that makes *substantial* contribution to an output neuron. In this section, we analyze the ERF size of an L -layers network with ARMA layers v.s. traditional convolutional layers. Formally, consider an output at location (i_1, i_2) , the impact from an input pixel at $(i_1 - p_1, i_2 - p_2)$ (i.e. L layers and (p_1, p_2) pixels away) is measured by the amplitude of partial derivative $g(i_1, i_2, p_1, p_2) = \left| \partial \mathcal{Y}_{i_1, i_2, t}^{(L)} / \partial \mathcal{X}_{i_1 - p_1, i_2 - p_2, s}^{(1)} \right|$ (where superscripts index the layers), i.e. how much the output changes as the input pixel is perturbed.

Definition 2 (Effective Receptive Field, ERF). Consider an L -layers network with an S -channels input $\mathcal{X}^{(1)} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and a T -channels output $\mathcal{Y}^{(L)} \in \mathbb{R}^{I_1 \times I_2 \times T}$, its effective receptive field is defined as the empirical distribution of the gradient maps: $ERF(p_1, p_2) = 1/(I_1 I_2 S T) \cdot \sum_{s, t, i_1, i_2} [g(i_1, i_2, p_1, p_2) / \sum_{j_1, j_2} g(j_1, j_2, p_1, p_2)]$. To measure the size of the ERF, we define its radius $r(ERF)$ as the standard deviation of the empirical distribution:

$$r^2(ERF) = \sum_{p_1, p_2} (p_1^2 + p_2^2) ERF(p_1, p_2) - \left[\sum_{p_1, p_2} \sqrt{p_1^2 + p_2^2} ERF(p_1, p_2) \right]^2 \quad (2)$$

Notice that ERF simultaneously depends on the model parameters and a specified input to the network, i.e. ERF is both *model-dependent* and *data-dependent*. Therefore, it is generally intractable to compute the ERF analytically for any practical neural network.

We follow the original paper of ERF [24] to estimate the radius with a simplified linear network. The paper empirically verifies that such an estimation is accurate and can be used to guide filter designs.

Theorem 3 (ERF of a linear ARMA network with dilated convolutions). Consider an L -layers linear network, where the ℓ^{th} layer computes $y_i^{(\ell)} - a^{(\ell)} y_{i-1}^{(\ell)} = \sum_{p=0}^{K^{(\ell)}-1} [(1-a^{(\ell)})/K^{(\ell)}] \cdot y_{i-d^{(\ell)}p}^{(\ell-1)}$ (i.e. the moving-average coefficients are uniform with length $K^{(\ell)}$ and dilation $d^{(\ell)}$, and the autoregressive coefficients $a_0^{(\ell)} = 1, a_1^{(\ell)} = -a^{(\ell)}$ has length 2). Suppose $0 \leq a^{(\ell)} < 1$ for $1 \leq \ell \leq L$, the ERF radius of such a linear ARMA network is

$$r(ERF)_{ARMA}^2 = \sum_{\ell=1}^L \left[\frac{d^{(\ell)2} (K^{(\ell)2} - 1)}{12} + \frac{a^{(\ell)}}{(1 - a^{(\ell)})^2} \right] \quad (3)$$

When $a^{(\ell)} = 0, \forall \ell \in [L]$, the ARMA layers reduce to (dilated) convolutional layers, and the ERF of the resulted linear CNN has radius $r(ERF)_{CNN}^2 = \sum_{\ell=1}^L d^{(\ell)2} (K^{(\ell)2} - 1)/12$.

Theorem 3 is proved in Appendix B. If the coefficients for different layers are identical, the radius reduces to $r(ERF)_{ARMA} = \sqrt{L} \cdot \sqrt{d^2(K^2 - 1)/12 + a/(1 - a)^2}$.

Remarks: (1) Compared with a (dilated) CNN, an ARMA network can have arbitrarily large ERF with an extra parameter a at each layer. When the autoregressive coefficient a is large (e.g. $a > 1 - 1/(dK)$), the second term $a/(1 - a)^2$ dominates the radius, and the ERF is substantially larger than that of a CNN. In particular, the radius tends to infinity as a approaches 1. **(2) An ARMA network can adaptively adjust its ERF through learnable parameter a .** As a gets smaller (e.g. $a < 1 - 1/(dK)$), the second term is comparable to or smaller than the first term, and the effect of expanded ERF diminishes. In particular if $a = 0$, an ARMA network reduces to a CNN.

Visualization of the ERF. In Theorem 3, we analytically show that the ARMA's radius of ERF increases with the network depth and magnitude of the autoregressive coefficients. We now verify our analysis by simulating linear ARMA networks with a single extra parameter (an autoregressive coefficient a) in each layer under varying depths and magnitude of the autoregressive coefficient a . Shown in Figure 3, as the autoregressive coefficient get larger, the radius of the ERF increases.

When the autoregressive coefficient is zero (i.e. $a = 0$), an ARMA network reduces to a traditional convolutional network. The simulation results also indicate that the ERF expands as the networks get deeper, and ARMA’s ability to expand the ERF increases as the networks get deeper. In conclusion, an ARMA network can have a large ERF even when the network is shallow, and its ability to expand the ERF increases as the network gets deeper.

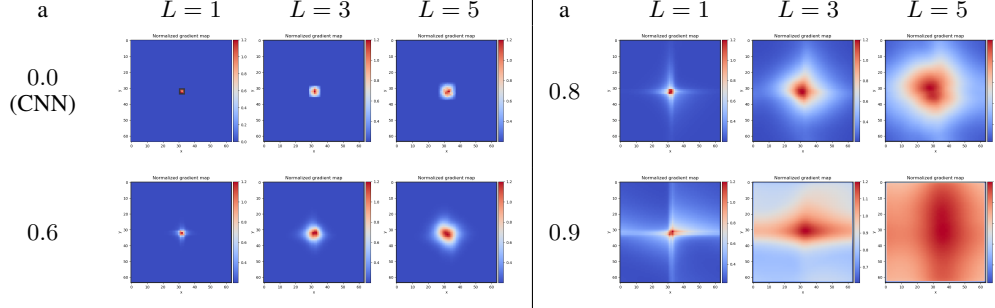


Figure 3: Visualization of ERF in linear ARMA networks with a single extra parameter (an autoregressive coefficient a) in each layer, under different network depth $L = 1, 3, 5$ and different magnitude of the autoregressive coefficient $a = 0.0, 0.6, 0.8, 0.9$.

4 Prediction and Learning of ARMA Layer

In the ARMA layer, each neuron is influenced by its neighbors from all directions (see Figure 2b). As a result, no neurons could be evaluated alone before evaluating any other neighboring neurons. To compute Equation 1, we thus need to solve a system of linear equations to obtain all values simultaneously. (1) However, the standard solver using Gaussian elimination is too expensive to be practical, and therefore we need to seek for a more efficient solution. (2) Furthermore, the solver for the system of linear equations is typically not automatic differentiable, and we have to derive the backward equations analytically. (3) Finally, we also need to devise an efficient algorithm to compute the backpropagation equations efficiently. In the section, we address these aforementioned problems.

Decomposing ARMA Layer. We decompose the ARMA layer in Equation 1 into a moving-average (MA) layer and an Autoregressive layer, with $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times T}$ as an intermediate result:

$$\text{MA Layer: } \mathcal{T}_{:, :, t} = \sum_{s=1}^S \mathcal{W}_{:, :, t, s} * \mathcal{X}_{:, :, s}; \quad \text{AR Layer: } \mathcal{A}_{:, :, t} * \mathcal{Y}_{:, :, t} = \mathcal{T}_{:, :, t} \quad (4)$$

Difficulty in Computing the AR Layer. While the MA layer is simply a traditional convolutional layer, it is nontrivial to solve the AR layer. Naively using Gaussian elimination, the linear equations in the AR layer can be solved in time cubic in dimension $O((I_1^2 + I_2^2)I_1 I_2 T)$, which is too expensive.

Solving the AR Layer. We propose to use the frequency-domain division [20] to solve the *deconvolution* problem in the AR layer. Since the convolution in spatial domain leads to element-wise product in frequency domain, we first transform \mathcal{A}, \mathcal{T} into their frequency representations $\tilde{\mathcal{A}}, \tilde{\mathcal{T}}$, with which we compute $\tilde{\mathcal{Y}}$ (the frequency representation of \mathcal{Y}) with element-wise division. Then, we reconstruct the output \mathcal{Y} by an inverse Fourier transform of $\tilde{\mathcal{Y}}$.

Computational Overhead. ARMA trades small overhead of extra number of parameters and computation for large gain of ERF radius as shown in Table 1. With *Fast Fourier Transform* (FFT), the FLOPS required by the extra autoregressive layer is $O(\log(\max(I_1, I_2))I_1 I_2 T)$ (see Appendix C for derivations). Importantly, compared with non-local attention block [33], the extra computation introduced in a ARMA layer is smaller; a non-local attention block requires $O(I_1^2 I_2^2 T)$ FLOPS.

Layer	# params.	# FLOPs	$r(\text{ERF})^2$
Conv.	$K_w^2 C^2$	$O(I^2 K_w^2 C^2)$	$O(LK_w^2)$
ARMA	$K_w^2 C^2 + K_a^2 C$	$O(K_w^2 I^2 C^2 + I^2 \log(I) C)$	$O(LK_w^2 + \frac{a}{(1-a)^2})$

Table 1: ARMA layer achieves large **gain** of ERF radius through small **overhead** of extra # of parameters and # of FLOPs. Through a single extra parameter a (thus $K_a = 2$), the ERF radius can be arbitrarily large. For notational simplicity, we assume the heights/widths are equal $I_1 = I_2 = I'_1 = I'_2 = I$, and the input and output channels are the same $S = T = C$.

Backpropagation. Deriving the backpropagation for Equation 4 is nontrivial; although backpropagation rule for MA layer is conventional, that of AR layer is not. In Theorem 4 we show that backpropagation of an AR layer can be computed as two ARMA models.

Theorem 4 (Backpropagation of ARMA layer). *Given $\mathcal{A}_{:,t} * \mathcal{Y}_{:,t} = \mathcal{T}_{:,t}$ and the gradient $\partial \mathcal{L} / \partial \mathcal{Y}$, the gradients $\{\partial \mathcal{L} / \partial \mathcal{A}, \partial \mathcal{L} / \partial \mathcal{X}\}$ can be obtained by two ARMA models:*

$$\mathcal{A}^\top_{:,t} * \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{:,t}} = -\mathcal{Y}^\top_{:,t} * \frac{\partial \mathcal{L}}{\partial \mathcal{Y}_{:,t}}; \quad \mathcal{A}^\top_{:,t} * \frac{\partial \mathcal{L}}{\partial \mathcal{T}_{:,t}} = \frac{\partial \mathcal{L}}{\partial \mathcal{Y}_{:,t}} \quad (5)$$

where $\mathcal{A}^\top_{:,t}$ and $\mathcal{Y}^\top_{:,t}$ are the transposed images of $\mathcal{A}_{:,t}$ and $\mathcal{Y}_{:,t}$ (e.g. $\mathcal{A}^\top_{i_1, i_2, t} = \mathcal{A}_{-i_1, -i_2, t}$).

Since the backpropagation is characterized by ARMA models, it can be evaluated efficiently using FFT similar to Equation 4. The proof of Theorem 4 with its FFT evaluation is given in Appendix C.

5 Stability of ARMA Layers

An ARMA model with arbitrary coefficients is not always stable. For example, the model $y_i - ay_{i-1} = x_i$ is unstable if $|a| > 1$: Consider an input \mathbf{x} with $x_0 = 1$ and $x_i = 0, \forall i \neq 0$, the output \mathbf{y} will recursively amplify itself as $y_0 = 1, y_1 = a, \dots, y_i = a^i$ and diverge to infinity.

5.1 Stability Constraints for ARMA layer

The key to guarantee stability of an ARMA layer is to constrain its autoregressive coefficients, which prevents the output from repeatedly amplifying itself. To derive the constraints, we propose a special design, *separable ARMA layer* inspired by *separable filters* [20].

Definition 5 (Separable ARMA Layer). *A separable ARMA layer is parameterized by a moving-average kernel $\mathcal{W} \in \mathbb{R}^{K_w \times K_w \times S \times T}$ and $T \times Q$ sets of autoregressive filters $\{(f_{:,t}^{(q)}, g_{:,t}^{(q)})_{q=1}^Q\}_{t=1}^T$. It takes an input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and returns an output $\mathcal{Y} \in \mathbb{R}^{I'_1 \times I'_2 \times T}$ as*

$$(f_{:,t}^{(1)} * \dots * f_{:,t}^{(Q)}) \otimes (g_{:,t}^{(1)} * \dots * g_{:,t}^{(Q)}) * \mathcal{Y}_{:,t} = \sum_{s=1}^S \mathcal{W}_{:,t,s} * \mathcal{X}_{:,t,s} \quad (6)$$

where the filters $f_{:,t}^{(q)}, g_{:,t}^{(q)} \in \mathbb{R}^3$ are length-3, and \otimes denotes outer product of two 1D-filters.

Remarks: Each autoregressive filter $\mathcal{A}_{:,t}$ is designed to be separable, i.e. $\mathcal{A}_{:,t} = F_{:,t} \otimes G_{:,t}$, thus it can be characterized by 1D-filters $F_{:,t}$ and $G_{:,t}$. By the fundamental theorem of algebra [28], any 1D-filter can be represented as a composition of length-3 filters. Therefore, $F_{:,t}$ and $G_{:,t}$ can further be factorized as $F_{:,t} = f_{:,t}^{(1)} * f_{:,t}^{(2)} * \dots * f_{:,t}^{(Q)}$ and $G_{:,t} = g_{:,t}^{(1)} * g_{:,t}^{(2)} * \dots * g_{:,t}^{(Q)}$. In summary, each $\mathcal{A}_{:,t}$ is characterized by Q sets of length-3 autoregressive filters $(f_{:,t}^{(q)}, g_{:,t}^{(q)})_{q=1}^Q$.

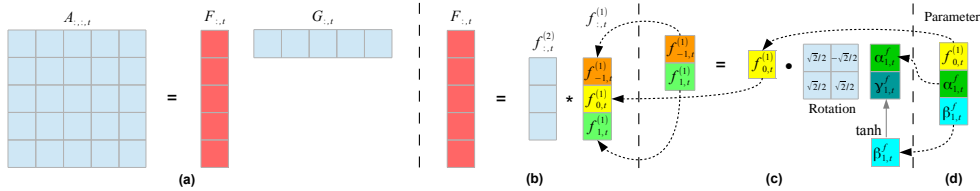


Figure 4: For each channel t , (a) the two-dimensional filter $\mathcal{A}_{:,t}$ is parameterized through an outer product of two 1D-filters $F_{:,t}$ and $G_{:,t}$; (b) $F_{:,t}$ is parameterized through a convolution of $f_{:,t}^{(1)} * \dots * f_{:,t}^{(Q)}$, and similarly $G_{:,t}$ as a convolution of $g_{:,t}^{(1)} * \dots * g_{:,t}^{(Q)}$; (c) we re-parameterize each constrained $(f_{:,t}^{(q)}, g_{:,t}^{(q)})$ to unconstrained $(\alpha_{q,t}^f, \beta_{q,t}^f)$, and similarly $(g_{:,t}^{(q)}, g_{:,t}^{(q)})$ to $(\alpha_{q,t}^g, \beta_{q,t}^g)$; (d) final parameters for unconstrained optimization are $(\alpha_{q,t}^f, \beta_{q,t}^f, \alpha_{q,t}^g, \beta_{q,t}^g)_{q=1}^Q$.

Theorem 6 (Constraints for Stable Separable ARMA Layer). *A sufficient condition for the separable ARMA layer (Definition 5) to be stable (i.e. output be bounded for any bounded input) is:*

$$|f_{-1,t}^{(q)} + f_{1,t}^{(q)}| < f_{0,t}^{(q)}, \quad |g_{-1,t}^{(q)} + g_{1,t}^{(q)}| < g_{0,t}^{(q)}, \quad \forall q \in [Q], t \in [T]. \quad (7)$$

The proof is deferred to Appendix D, which follows the standard techniques using Z-transform.

5.2 Achieving stability via re-parameterization

In principle, the constraints required for stability in a ARMA layer as in Theorem 6 could be enforced through constrained optimization. However, constrained optimization algorithm, such as projected gradient descent [2], is more expensive as it requires an extra projection step. Moreover, it could be more difficult to achieve convergence. In order to avoid the aforementioned challenges, we introduce a *re-parameterization* mechanism to remove constraints needed to guarantee stability in ARMA layer.

Theorem 7 (Re-parameterization). *For a separable ARMA layer in Definition 5, if we re-parameterize each tuple $(f_{-1,t}^{(q)}, f_{1,t}^{(q)}, g_{-1,t}^{(q)}, g_{1,t}^{(q)})$ as learnable parameters $(\alpha_{q,t}^f, \beta_{q,t}^f, \alpha_{q,t}^g, \beta_{q,t}^g)$:*

$$\begin{pmatrix} f_{-1,t}^{(q)} & g_{-1,t}^{(q)} \\ f_{1,t}^{(q)} & g_{1,t}^{(q)} \end{pmatrix} = \begin{pmatrix} f_{0,t}^{(q)} & 0 \\ 0 & g_{0,t}^{(q)} \end{pmatrix} \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} \begin{pmatrix} \alpha_{q,t}^f & \alpha_{q,t}^g \\ \tanh(\beta_{q,t}^f) & \tanh(\beta_{q,t}^g) \end{pmatrix} \quad (8)$$

then the layer is stable for arbitrary $\{(f_{0,t}^{(q)}, g_{0,t}^{(q)}, \alpha_{q,t}^f, \beta_{q,t}^f, \alpha_{q,t}^g, \beta_{q,t}^g)_{q=1}^Q\}_{t=1}^T$ with no constraints.

In practice, we can set $f_{0,t}^q = g_{0,t}^q = 1$ (since the scale can be learned by the moving-average kernel), and only store and optimize over each tuple $(\alpha_{q,t}^f, \beta_{q,t}^f, \alpha_{q,t}^g, \beta_{q,t}^g)$. In other words, each autoregressive filter $\mathcal{A}_{:,t}$ is constructed from $(\alpha_{q,t}^f, \beta_{q,t}^f, \alpha_{q,t}^g, \beta_{q,t}^g)_{q=1}^Q$ on the fly during training or inference.

Experimental Demonstration of Re-parameterization. To verify the re-parameterization is essential for stable training, we train a VGG-11 network [30] on CIFAR-10 dataset, where all convolutional layers are replaced by ARMA layers with autoregressive coefficients initialized as zeros. We compare the learning curves using the re-parameterization v.s. not using the re-parameterization in Figure 5. As we can see, the training quickly converges under our proposed re-parameterization mechanism with which the stability of the network is guaranteed. However without the re-parameterization mechanism, a naive training of ARMA network never converges and gets NaN error quickly. The experiment thus verifies that the theory in Theorem 7 is effective in guaranteeing stability.

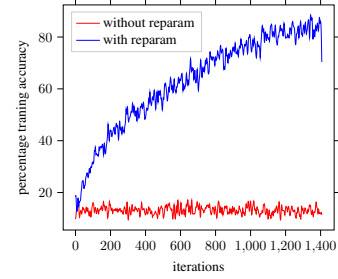


Figure 5: Learning curves with and without re-parameterization on an ARMA network with a VGG-11 backbone on CIFAR-10.

6 Experiments

We apply our ARMA networks on two dense prediction problems – pixel-level video prediction and semantic segmentation to demonstrate effectiveness of ARMA networks. **(1)** We incorporate our ARMA layers in U-Net [29, 35] for semantic segmentation, and in ConvLSTM network [5, 38] for video prediction. **We show that the resulted ARMA U-Net and ARMA-LSTM models uniformly outperform the baselines on both tasks.** **(2)** We then interpret the varying performance of ARMA networks on different tasks by visualizing the histograms of the learned autoregressive coefficients. We include the detailed setups (datasets, model architectures, training strategies and evaluation metrics) and visualization in Appendix A for reproducibility purposes.

Semantic Segmentation on Biomedical Medical Images. We evaluate our ARMA U-Net on the lesion segmentation task in ISIC 2018 challenge [37], comparing against a baseline U-Net [29] and non-local U-Net [35] (U-Net augmented with non-local attention blocks).

Table 2: Semantic segmentation on ISIC dataset. For all metrics (ACC, SE, SP, PC, F1 and JS), higher values indicates better performance. The reported numbers are an average of 10 runs with different seeds.

Model	params.	ACC	SE	SP	PC	F1	JS
U-Net [29]	3.453M	0.946 ± 0.003	0.884 ± 0.019	0.977 ± 0.005	0.857 ± 0.020	0.842 ± 0.009	0.754 ± 0.011
NL U-Net [35]	4.403M	0.945 ± 0.003	0.877 ± 0.017	0.973 ± 0.004	0.844 ± 0.014	0.831 ± 0.012	0.741 ± 0.013
ARMA U-Net	3.455M	0.955 ± 0.003	0.896 ± 0.011	0.972 ± 0.005	0.873 ± 0.011	0.861 ± 0.007	0.780 ± 0.009
NL ARMA U-Net	4.405M	0.960 ± 0.002	0.909 ± 0.009	0.968 ± 0.004	0.870 ± 0.011	0.870 ± 0.006	0.790 ± 0.008

ARMA networks outperform both baselines in almost all metrics. As shown in Table 2, our (non-local) ARMA U-Net outperform both U-Net and non-local U-Net except for specificity (SP). Furthermore, we find that the synergy of non-local attention and ARMA layers achieves best results among all.

Pixel-level Video Prediction. We evaluate our ARMA-LSTM network on the Moving-MNIST-2 dataset [11] with different moving velocities, comparing against the baseline ConvLSTM network [5, 38] and its augmentation using dilated convolutions and non-local attention blocks [33]. As shown in the visualizations in Appendix A, the dilated ARMA-LSTM does not have gridding artifacts as in dilated Conv-LSTM, that is *ARMA removes the gridding artifacts*.

Table 3: 10-frames **video prediction** on Moving-MNIST-2 with three different speeds (results averaged over 10 predicted frames). MA and AR denote the size of moving-average and autoregressive kernels respectively, and dil. denotes dilation in the moving-average kernel. Higher PSNR, SSIM values indicate better performance.

Model	MA	AR	dil.	params.	original speed		2X speed		3X speed	
					PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Conv-LSTM (size 3)	3	1	1	0.887M	18.24	0.867	16.62	0.827	15.81	0.810
Conv-LSTM (size 5)	5	1	1	2.462M	19.58	0.901	17.61	0.856	16.99	0.841
Dilated Conv-LSTM	3	2	2	0.887M	19.16	0.893	17.92	0.858	17.48	0.846
Dilated ARMA-LSTM	3	3	2	0.893M	19.72	0.904	18.05	0.870	17.65	0.855
ARMA-LSTM (size 3)	3	2	1	0.893M	19.72	0.899	18.73	0.881	18.13	0.869

ARMA networks outperforms larger networks: As shown in Table 3, our ARMA networks with kernel sizes 3×3 outperform all baselines under all velocities (at the original speed, our ARMA network requires dilated convolutions to achieve the best performance). Moreover, for videos with a higher moving speed, the advantage is more pronounced as expected due to ARMA’s ability to expand the ERF. The ARMA networks improve the best baseline (Conv-LSTM with kernel size 5×5) in PSNR by 6.36% at 2X speed and by 6.70% at 3X speed, with 63.7% fewer parameters.

ARMA networks outperforms non-local attention blocks: As shown in Table 4, our ARMA-LSTM with kernel sizes 3×3 outperforms the Conv-LSTMs augmented by non-local attention blocks. However, attention mechanism does not always improve the baselines or our models. When both ARMA-LSTM and Conv-LSTM are combined with non-local attention blocks, our model achieves better performance compared to the non-ARMA baselines.

Table 4: Comparison with non-local attention blocks on **video prediction**. The original networks are the same as in Table 3. Each non-local network additionally inserts two non-local blocks in the corresponding base network.

Model	Original		Non-local	
	PSNR	SSIM	PSNR	SSIM
ConvLSTM (size 3)	18.24	0.867	19.45	0.895
ConvLSTM (size 5)	19.58	0.901	19.18	0.891
ARMA-LSTM (size 3)	19.72	0.899	19.62	0.897

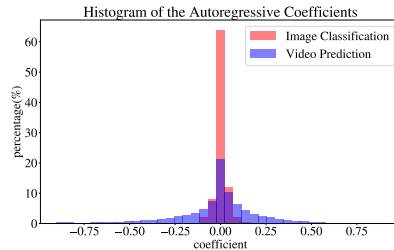


Figure 6: Histogram of the autoregressive coefficients in trained ARMA networks.

Interpretation by Autoregressive Coefficients. To explain why ARMA networks achieve impressive performance in dense prediction, Figure 6 compares the histograms of the trained autoregressive coefficients between video prediction and image classification. (subsection A.4 demonstrates performance of image classifications when ARMA layers are incorporated in VGG and ResNet.)

1. The histograms demonstrate how ARMA networks adaptively learn autoregressive coefficients according to the tasks. As motivated in the introduction, dense prediction such as video prediction requires each layer to have a large receptive field such that global information is captured.
2. The large autoregressive coefficients in video prediction model suggests the overall ERF is significantly expanded. In image classification model, global information is already aggregated by pooling (downsampling) layers and a fully-connected classification layer. Therefore, the ARMA layers automatically learn nearly zero autoregressive coefficients.

7 Discussion

This paper proposes a novel *ARMA* layer capable of expanding a network’s effective receptive field adaptively. Our method is related to techniques in signal processing and machine learning.

First, ARMA layer is equivalent to a multi-channel *impulse response filter* in signal processing [28]. Alternatively, we can interpret the autoregressive layer as a learnable *spectral normalization* [25] following the moving-average layer. Additionally, the ARMA layer is an *linear recurrent neural network*, where the recurrent propagations are over the spatial domain (section 2).

References

- [1] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.
- [2] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.
- [3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [4] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555, 2015.
- [5] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Contextvp: Fully context-aware video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 753–769, 2018.
- [6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [8] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.
- [9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.
- [10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.
- [11] Github repo. https://github.com/jthsieh/DDPAE-video-prediction/blob/master/data/moving_mnist.py. [Online; accessed 05-Jan-2020].
- [12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [14] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990.

- [15] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4201–4209, 2017.
- [16] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.
- [17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [19] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6054–6063, 2019.
- [20] Jae S Lim. Two-dimensional signal and image processing. *ph*, 1990.
- [21] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems*, pages 1520–1530, 2017.
- [22] Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, pages 560–576. Springer, 2016.
- [23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [24] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pages 4898–4906, 2016.
- [25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.
- [26] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.
- [27] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.
- [28] Alan V Oppenheim, John R Buck, and Ronald W Schafer. Discrete-time signal processing. 2014.
- [29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [31] Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in neural information processing systems*, pages 2998–3006, 2015.
- [32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

- [33] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.
- [34] Zhengyang Wang and Shuiwang Ji. Smoothed dilated convolutions for improved dense prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2486–2495, 2018.
- [35] Zhengyang Wang, Na Zou, Dinggang Shen, and Shuiwang Ji. Non-local u-nets for biomedical image segmentation. In *AAAI*, pages 6315–6322, 2020.
- [36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
- [37] Webpage. <https://challenge2018.isic-archive.com>. [Online; accessed 29-May-2020].
- [38] SHI Xingjian, Zhouong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.
- [39] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [40] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.
- [41] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019.

Appendix of ARMA Nets: Expanding Receptive Field for Dense Prediction

A Supplementary Materials for Experiments

In this section, we explain detailed setups (datasets, model architectures, learning strategies and evaluation metrics) of all experiments, and provide additional visualizations of the results.

A.1 Visualization of Effective Receptive Field

In the simulations in subsection 3.2, all linear networks have 32 channels and 64×64 feature size at each layer. The filter size for both moving-average coefficients and autoregressive coefficients is set to 3×3 : each moving-average kernel \mathcal{W} is initialized using Xavier’s method, while the autoregressive kernel \mathcal{A} is initialized randomly within a stable region $-a \leq f_{-1,t}^{(q)} + f_{1,t}^{(q)} \leq 0$, $-a \leq g_{-1,t}^{(q)} + g_{1,t}^{(q)} \leq 0$, $\forall t \in [T]$, $q \in [Q]$ (see section 5 for details). Each heat map in Figure 3 is computed as an average of 32 gradient maps from different channels.

A.2 Multi-frame Video Prediction

Datasets and Metrics The Moving-MNIST-2 dataset is generated by moving two digits of size 28×28 in MNIST dataset within a 64×64 black canvas [11]. These digits are placed at a random initial location, and move with constant velocity in the canvas and bounce when they reach the boundary. In addition to the default velocity in the public generator [11], we increase the velocity to $2\times$ and $3\times$ to test all models on videos with stronger motions. For each velocity, we generate 10,000 videos for training set, 3,000 for validation set, and 5,000 for test set, where each video contains 20 frames. All models are trained to the next 10 frames given 10 input frames, and we evaluate their performance based on the metrics of *mean square error* (MSE), *peak signal-noise ratio* (PSNR) and *structure similarity* (SSIM) [36].

Model Architectures (1) **Baselines.** The backbone architecture consists of a stack of 12 Conv-LSTM modules, and each module contains 32 units (channels). Following [5], two skip connections that perform channel concatenation are added between (3, 9) and (6, 12) module. An additional traditional convolutional layer is applied on top of all recurrent layers to compute the predicted frames. The backbone architecture is illustrated in Figure 7. In the baseline networks, we consider three different convolutions at each layer: (a) Traditional convolution with filter size 3×3 ; (b) Traditional convolution with filter size 5×5 ; and (c) 2-dilated convolution with filter size 3×3 .

(2) **ARMA networks.** Our ARMA networks use the same backbone architecture as baselines, but replace their convolutional layers with ARMA layers. For all ARMA models, we set the filter size for both moving-average and autoregressive parts to 3×3 . In the ARMA networks, we consider two different convolutions each layer: (a) The moving-average part is a traditional convolution; (b) We further consider using 2-dilated convolution in the moving-average part.

(3) **Non-local networks.** In non-local networks, we additionally insert two non-local block in the backbone architecture, as illustrated in Figure 8. In each non-local block, we use embedded Gaussian as the non-local operation [33], and we replace the batch normalization by instance normalization that is compatible to recurrent neural networks. In non-local networks, we consider three types of convolutions at each layer: (1)(2) Traditional convolutions with filter size 3×3 and 5×5 ; (3) ARMA layer with 3×3 moving-average and autoregressive filters.

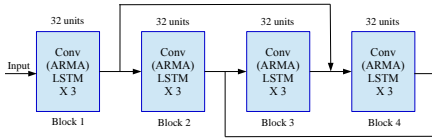


Figure 7: Conv(ARMA)-LSTM.

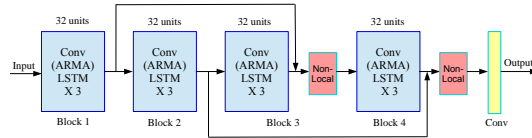


Figure 8: Non-Local Conv(ARMA)-LSTM.

Training Strategy All models are trained using ADAM optimizer [17], with $\mathcal{L}_1 + \mathcal{L}_2$ loss and for 500 epochs. We set the initial learning rate to 10^{-3} , and the value for gradient clipping to 3. Learning

rate decay and scheduled sampling [1] are used to ease training. Scheduled sampling is started once the model does not improve in 20 epochs (in term of validation loss), and the sampling ratio is decreased linearly by 4×10^{-4} each epoch (i.e. scheduling sampling lasts for 250 epochs). Learning rate decay is further activated if the validation loss does not drop in 20 epochs, and the learning rate is decreased exponentially by 0.98 every 5 epochs. All convolutional layers and moving-average parts in ARMA layers are initialized by Xavier’s normalized initializer [12], and autoregressive coefficients in ARMA layers are initialized as zeros (i.e. each ARMA layer is initialized as a traditional layer).

Visualization of the Predictions We visualize the predictions by different models under three moving velocities in Figure 9, Figure 10 and Figure 11. Notice that the gridding artifacts by dilated convolutions are removed by ARMA layer: since each neuron receives information from all pixels in a local region (Figure 2d), adjacent neurons are no longer computed from separate sets of pixels. Moreover, for videos with a higher moving speed, the advantage of our ARMA layer is more pronounced as expected due to ARMA’s ability to expand the ERF.

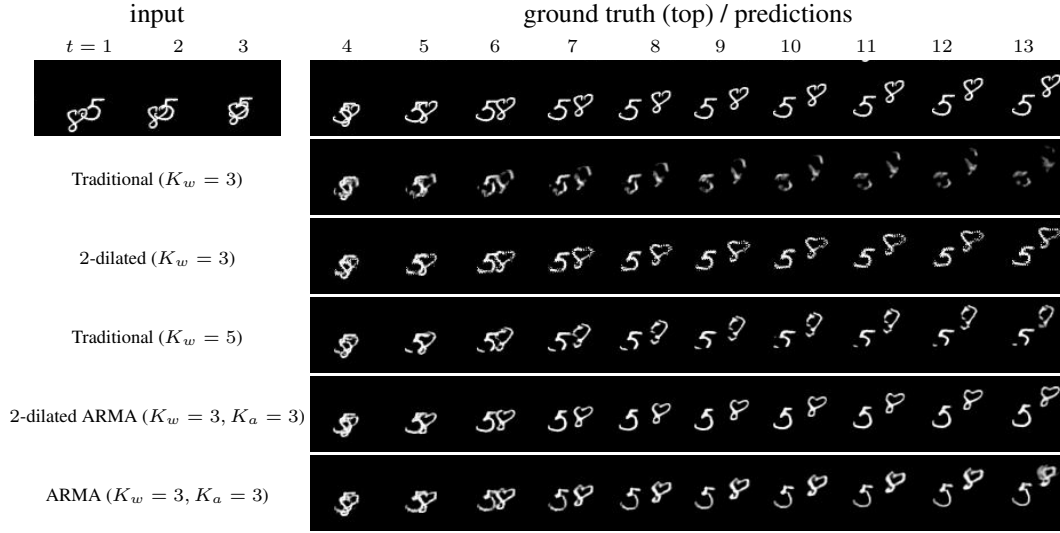


Figure 9: Prediction on Moving-MNIST-2 (original speed). The first row contains the last 3 input frames and 10 ground-truth frames for models to predict.

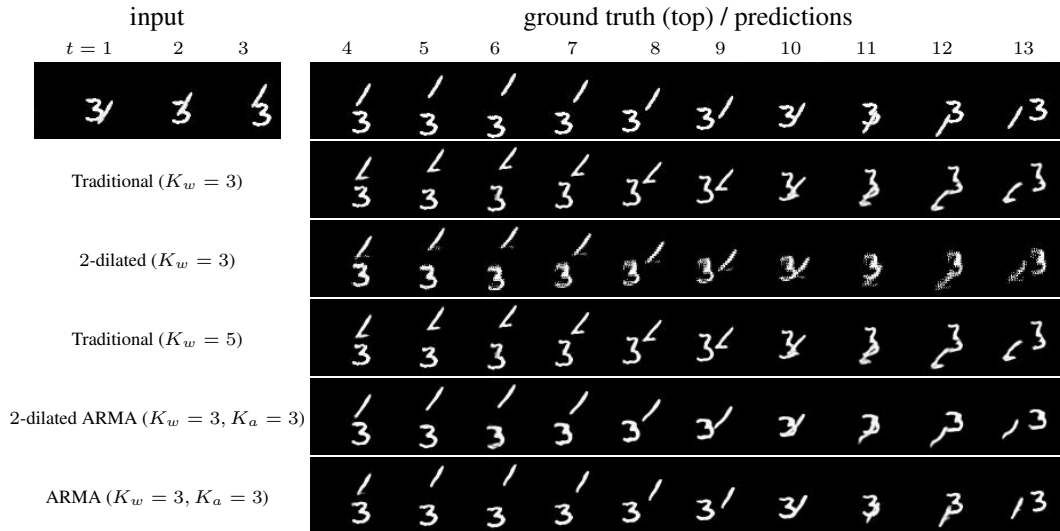


Figure 10: Prediction on Moving-MNIST-2 (2× speed). The first row contains the last 3 input frames and 10 ground-truth frames for models to predict.

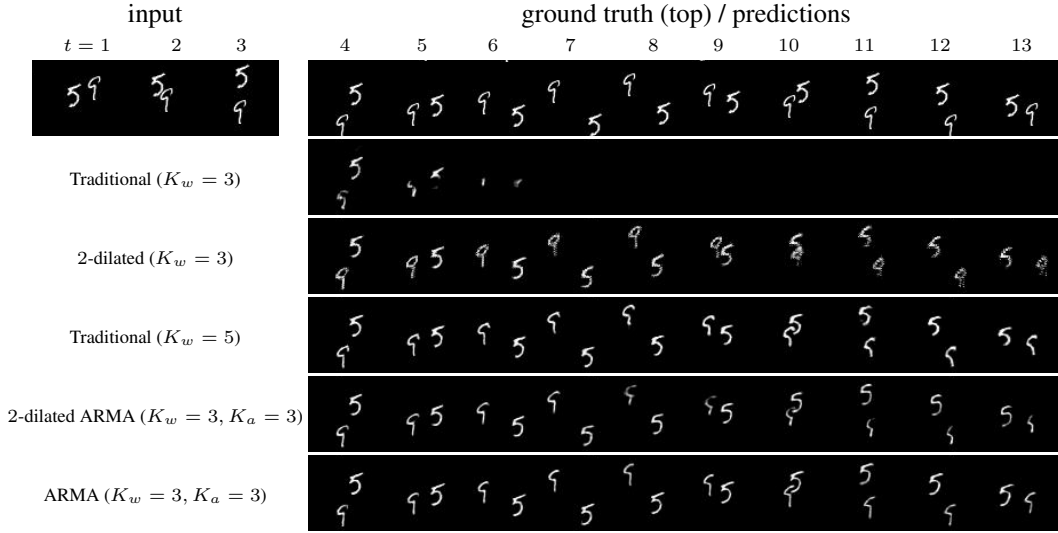


Figure 11: Prediction on Moving-MNIST-2 ($3\times$ speed). The first row contains the last 3 input frames and 10 ground-truth frames for models to predict.

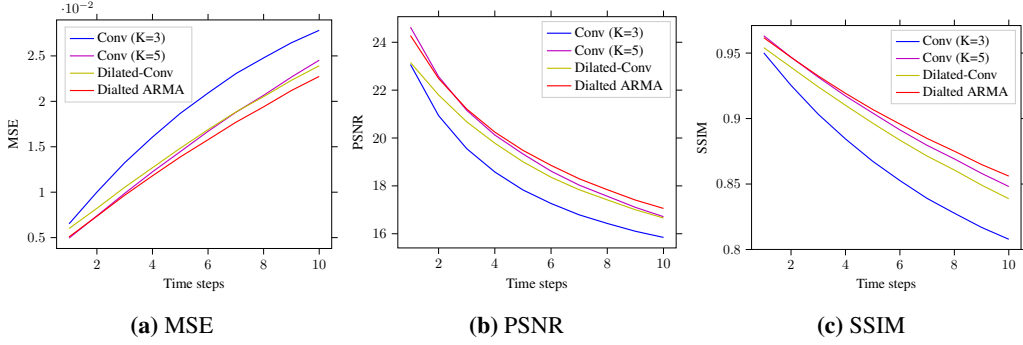


Figure 12: Per-frame performance comparison of our ARMA and our dilated ARMA networks v.s. the Conv-LSTM, dilated Conv-LSTM baselines for Moving-MNIST-2 (original speed). Lower MSE values (in 10^{-3}) or higher PSNR/SSIM values indicate better performance.

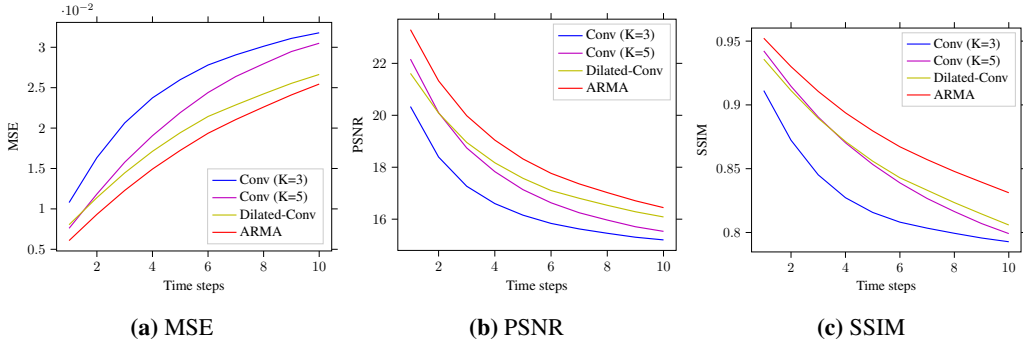


Figure 13: Per-frame performance comparison of our ARMA and our dilated ARMA networks v.s. the Conv-LSTM, dilated Conv-LSTM baselines for Moving-MNIST-2 ($2\times$ speed). Lower MSE values (in 10^{-3}) or higher PSNR/SSIM values indicate better performance.

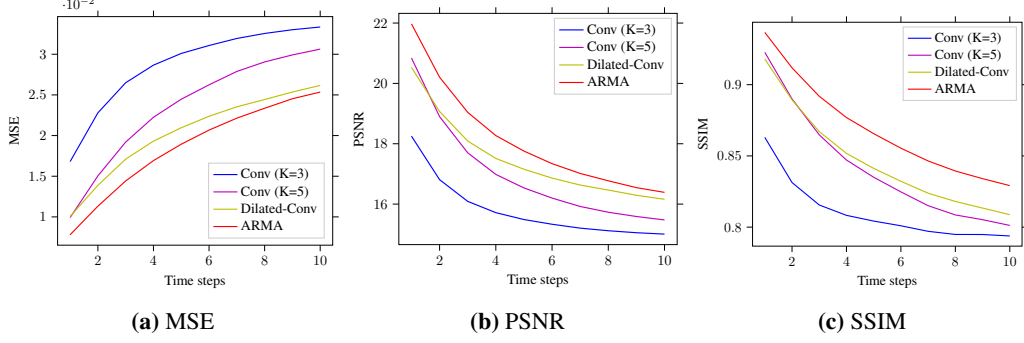


Figure 14: Per-frame performance comparison of our ARMA and our dilated ARMA networks v.s. the Conv-LSTM, dilated Conv-LSTM baselines for Moving-MNIST-2 ($3\times$ speed). Lower MSE values (in 10^{-3}) or higher PSNR/SSIM values indicate better performance.

A.3 Medical Image Segmentation

To demonstrate ARMA networks’ applicability to image segmentation, we evaluate it on a challenging medical image segmentation problem.

Dataset and Metrics For all experiments, we use a dataset from ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection [8], which can be downloaded online¹. In this task, a model aims to predict a binary mask that indicates the location of the primary skin lesion for each input image. The dataset consists of 2594 images, and we resize each image to 224×224 . We split dataset into training set, validation set and test set with ratios of 0.7, 0.1 and 0.2 respectively. All models are evaluated using the following metrics: $AC = (TP + TN)/(TP + TN + FP + FN)$, $SE = TP/(TP + FN)$, $SP = TN/(TN + FP)$, $PC = TP/(TP + FP)$, $F1 = 2PC \cdot SE/(PC + SE)$ and $JS = |GT \cap SR|/|GT \cup SR|$, where TP stands for true positive, TN for true negative, FP for false positive, FN for false negative, GT for ground truth mask and SR for predictive mask.

Model Architectures (1) **Baselines.** We use U-Net [29] and non-local U-Net [35] as baseline models. U-Net has a contracting path to capture context and a symmetric expanding path enables precise localization. The network architecture is illustrated in Figure 15a. Non-local U-Net is equipped with global aggregation blocks based on the self-attention operator to aggregate global information without a deep encoder for biomedical image segmentation, which is illustrated in Figure 15b. (2) **Our architectures.** We replace all traditional convolution layers with ARMA layers in U-Net and non-local U-Net.

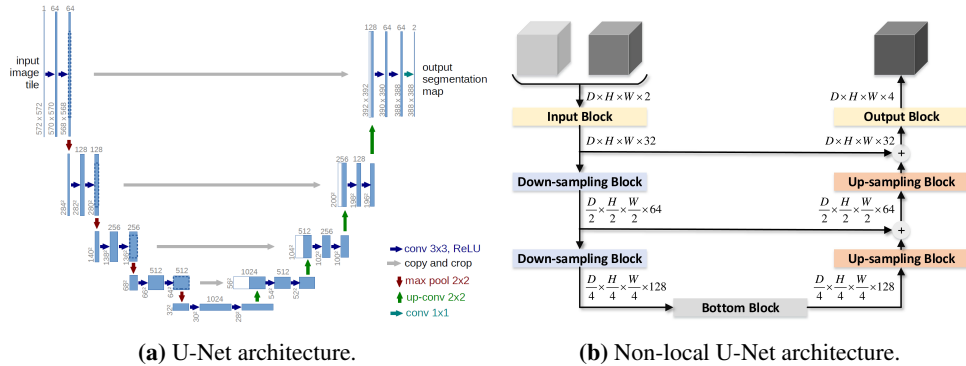


Figure 15: Backbone architectures.

Training Strategy. All models are trained using ADAM optimizer [17] with binary cross entropy (BCE) loss. For initial learning rate, we search from 10^{-2} to 10^{-5} and choose 10^{-3} for U-Net and

¹<https://challenge2018.isic-archive.com/task1/training/>

10^{-2} for non-local U-Net. The learning rate is decayed by 0.98 every epoch. During training, each image is randomly augmented by rotation, cropping, shifting, color jitter and normalization following the public source code².

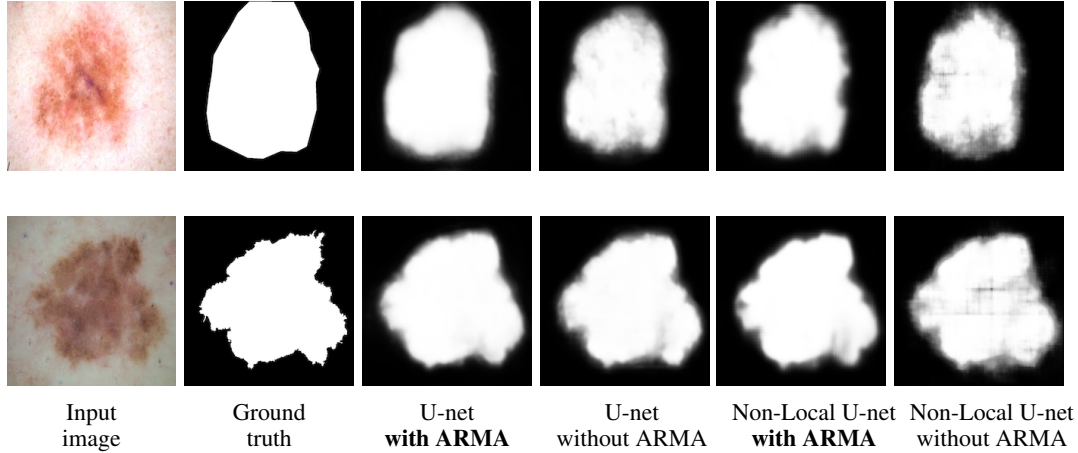


Figure 16: Predictive results of U-Net and Non-local U-Net with/without ARMA layers.

A.4 Image Classification

Model Architectures and Datasets We replace the traditional convolutional layers by ARMA layers in three benchmarking architectures for image classification: AlexNet [18], VGG-11 [30], and ResNet-18 [13]. We apply our proposed ARMA networks on CIFAR10 and CIFAR100 datasets. Both datasets have 50000 training examples and 10000 test examples, and we use 5000 examples from the training set for validation (and leave 45000 examples for training).

Training Strategy All models are trained using cross-entropy loss and SGD optimizer with batch size 128, learning rate 0.1, weight decay 0.0005 and momentum 0.9. For CIFAR10, the models are trained for 300 epochs and we half the learning rate every 30 epochs. For CIFAR100, the models are trained for 200 epochs and we divide the learning rate by 5 at the 60th, 120th, 160th epochs.

Results The experimental results are summarized in Table 5. Our results show that ARMA models achieve comparable or slightly better results than the benchmarking architectures. Replacing traditional convolutional layer with our proposed ARMA layer slightly boosts the performance of VGG-11 and ResNet-18 by 0.01%-0.1% in accuracy. **Since image classifications tasks do not require convolutional layers to have large receptive field, the learned autoregressive coefficients are highly concentrated around 0 as shown in Figure 6.** Consequently, ARMA networks effectively reduce to traditional convolutional neural networks and therefore achieve comparable results.

	AlexNet		VGG-11		ResNet-18	
	Conv.	ARMA	Conv.	ARMA	Conv.	ARMA
CIFAR10	86.30 \pm 0.29	85.67 \pm 0.19	91.57 \pm 0.59	91.57 \pm 0.73	95.01 \pm 0.15	95.07 \pm 0.13
CIFAR100	58.99 \pm 0.37	57.43 \pm 0.24	68.25 \pm 0.11	68.36 \pm 1.67	73.71 \pm 0.23	73.72 \pm 0.52

Table 5: Image classification on CIFAR10 and CIFAR100. The reported accuracies (%) and their standard deviations are computed from 10 runs with different seeds. Since image classifications tasks do not require convolutional layers to have large receptive field, the learned autoregressive coefficients are highly concentrated around 0 as shown in Figure 6. Consequently, ARMA networks effectively reduce to traditional CNNs and therefore achieve comparable results.

²https://github.com/LeeJunHyun/Image_Segmentation

B Analysis of Effective Receptive Field (ERF)

In this section, we prove the Theorem 3 in subsection 3.2. Throughout this section, we use $\mathbf{a}^{(\ell)} = \{\dots, a_{-1}^{(\ell)}, a_0^{(\ell)}, a_1^{(\ell)}, \dots\}$ to denote the ℓ^{th} layer's autoregressive coefficients, and $\mathbf{w}^{(\ell)} = \{\dots, w_{-1}^{(\ell)}, w_0^{(\ell)}, w_1^{(\ell)}, \dots\}$ the ℓ^{th} layer's autoregressive coefficients.

B.1 ERF of General Linear Convolutional Networks

The proof of is Theorem 3 based on the following theorem on linear convolutional networks [24], which includes both CNN and ARMA networks as special cases.

Theorem 8 (ERF of linear convolutional networks with infinite horizon). *Consider an L -layer linear convolutional network (without activation and pooling), where its ℓ^{th} -layer computes a weighted-sum of its input $y_i^{(\ell)} = \sum_{p=-\infty}^{+\infty} w_p^{(\ell)} y_{i-p}^{(\ell-1)}$. Suppose the weights are non-negative $w_p^{(\ell)} \geq 0$ and normalized at each layer $\sum_{p=-\infty}^{+\infty} w_p^{(\ell)} = 1, 1 \leq \ell \leq [L]$, the network has an ERF radius as*

$$r^2(\text{ERF}) = \sum_{\ell=1}^L \left[\sum_{p=-\infty}^{+\infty} p^2 w_p^{(\ell)} - \left(\sum_{p=-\infty}^{+\infty} p w_p^{(\ell)} \right)^2 \right] \quad (\text{B.1})$$

Furthermore, the ERF converges to a Gaussian density function when L tends to infinity.

Proof. In this linear convolutional network, the gradient maps can be computed with chain rule as

$$g_{i,:} = \mathbf{w}^{(1)\top} * \mathbf{w}^{(2)\top} \dots * \mathbf{w}^{(L)\top}, \forall i \in \mathbb{Z} \quad (\text{B.2})$$

where $\mathbf{w}^{\ell\top}$ denotes the reversed sequence of $\mathbf{w}^{(\ell)}$. Notice that (1) The gradient maps do not depend on the input, i.e. they are data-independent; (2) The gradient maps are identical across different locations in the output. Consequently, the ERF is equal to any one gradient map above

$$\text{ERF} = \mathbf{w}^{(1)\top} * \mathbf{w}^{(2)\top} \dots * \mathbf{w}^{(L)\top} \quad (\text{B.3})$$

The remaining part of the proof makes use of *probabilistic method*, which interprets the operation at each layer as a discrete random variable. Since the weights at each layer are non-negative and normalized, they can be treated as values of a probability mass function. Concretely, we construct L independent random variables $\{W^{(1)}, \dots, W^{(L)}\}$ such that $\mathbb{P}[W^{(\ell)} = p] = w_p^{(\ell)}$. Similarly, we introduce a random variable $S^{(L)}$ to represent the ERF, i.e. $\mathbb{P}[S^{(L)} = p] = \text{ERF}_p$. As a result, the ERF radius is equal to standard deviation of $S^{(L)}$, or equivalently $r^2(\text{ERF}) = \mathbb{V}[S^{(L)}]$.

Recall that *addition of independent random variables results in convolution of their probability mass functions*, Equation B.3 implies that $S^{(L)}$ is an addition of all $W^{(\ell)}$'s, i.e. $S^{(L)} = \sum_{\ell=1}^L W^{(\ell)}$. Therefore, the variance of $S^{(L)}$ is equal to a summation of the variances for $W^{(\ell)}$'s. Thus,

$$\mathbb{V}[S^{(L)}] = \sum_{\ell=1}^L \mathbb{V}[W^{(\ell)}] = \sum_{\ell=1}^L \left[\mathbb{E}[(W^{(\ell)})^2] - \mathbb{E}[W^{(\ell)}]^2 \right] \quad (\text{B.4})$$

$$= \sum_{\ell=1}^L \left[\sum_{p=-\infty}^{+\infty} p^2 w_p^{(\ell)} - \left(\sum_{p=-\infty}^{+\infty} p w_p^{(\ell)} \right)^2 \right] \quad (\text{B.5})$$

which proves the Equation B.1. Furthermore, the *Lyapunov central limit theorem* shows that $(S^{(L)} - \mathbb{E}[S^{(L)}])/\sqrt{\mathbb{V}[S^{(L)}]}$ converges to a standard normal random variable if L tends to infinity

$$\frac{S^{(L)} - \mathbb{E}[S^{(L)}]}{\sqrt{\mathbb{V}[S^{(L)}]}} = \frac{\sum_{\ell=1}^L (W^{(\ell)} - \mathbb{E}[W^{(\ell)}])}{\sqrt{\sum_{\ell=1}^L \mathbb{V}[W^{(\ell)}]}} \xrightarrow{D} \mathcal{N}(0, 1) \quad (\text{B.6})$$

that is, the ERF function is approximately Gaussian when the number of layers L is large enough. \square

B.2 ERF of Traditional CNNs ($a_1^{(\ell)} = -a^{(\ell)} = 0$)

As a warmup, we first provide a proof for the special case of traditional CNN where $a^{(\ell)} = 0$ for all layers. For reference, we list the first two cases of *Faulhaber's formula*:

$$\sum_{p=0}^{K-1} p = \frac{K(K-1)}{2} \quad (\text{B.7a})$$

$$\sum_{p=0}^{K-1} p^2 = \frac{K(K-1)(2K-1)}{6} \quad (\text{B.7b})$$

Proof. In this special case, the ERF radius can be obtained by plugging $w_p^{(\ell)} = 1/K^{(\ell)}$ for $p = 0, d^{(\ell)}, \dots, d^{(\ell)}(K^{(\ell)} - 1)$ into Equation B.1.

$$r^2(\text{ERF}) = \sum_{\ell=1}^L \left[\sum_{p=0}^{K^{(\ell)}-1} \frac{(pd^{(\ell)})^2}{K^{(\ell)}} - \left(\sum_{p=0}^{K^{(\ell)}-1} \frac{pd^{(\ell)}}{K^{(\ell)}} \right)^2 \right] \quad (\text{B.8})$$

$$= \sum_{\ell=1}^L \left[\frac{d^{(\ell)2} K^{(\ell)} (K^{(\ell)} - 1) (2K^{(\ell)} - 1)}{6} - \left(\frac{d^{(\ell)} K^{(\ell)} (K^{(\ell)} - 1)}{2} \right)^2 \right] \quad (\text{B.9})$$

$$= \sum_{\ell=1}^L \frac{d^{(\ell)2} (K^{(\ell)2} - 1)}{12} \quad (\text{B.10})$$

where the infinite series are computed using Equation B.7a and Equation B.7b. Taking square root on both sides completes the proof for the special case of CNNs. \square

ERF Analysis of CNNs. If we further assume that all layers are identical, i.e. $K^{(\ell)} = K, d^{(\ell)} = d$ for $1 \leq \ell \leq L$, we can simplify Equation B.10 as

$$r(\text{ERF}) = \sqrt{L} \cdot \sqrt{\frac{d^2(K^2 - 1)}{12}} = O(dK\sqrt{L}) \quad (\text{B.11})$$

That is, the ERF radius grows linearly with kernel size K and dilation d , but sub-linearly with the number of layers L in the linear network.

B.3 ERF of ARMA networks ($a_1^{(\ell)} = -a^{(\ell)} \leq 0$)

In the part, we provide a proof for general ARMA networks where $a^{(\ell)} \leq 0$. In sketch, the proof consists of three steps: **(1)** we introduce *inverse convolution* and convert each ARMA model to a moving-average model: $\mathbf{a} * \mathbf{y} = \mathbf{w} * \mathbf{x} \implies \mathbf{y} = \mathbf{f} * \mathbf{x}$, where \mathbf{f} represents a convolution with *infinite* number of coefficients, \mathbf{x} and \mathbf{y} are the input and output of the model respectively. **(2)** We derive the *moment generating function* (MGF) of the moving-average coefficients from the first step, and use the functions to compute the first and second moments. **(3)** We plug the moments from the second step into Equation B.1 to obtain Equation 3.

Definition 9 (Inverse convolution). Given a convolution (with coefficients) \mathbf{a} , its inverse convolution $\bar{\mathbf{a}}$ is defined such that $\mathbf{a} * \bar{\mathbf{a}} = \bar{\mathbf{a}} * \mathbf{a} = \delta$ is an identical mapping, i.e.

$$\sum_{p=-\infty}^{+\infty} a_{i-p} \bar{a}_p = \delta_i = \begin{cases} 1 & i = 0 \\ 0 & i \neq 0 \end{cases} \quad (\text{B.12})$$

Remark: The inverse convolution does not exist for any convolution \mathbf{a} . A necessary and sufficient condition for invertibility of \mathbf{a} is that its Fourier transform is non-zero everywhere [28].

Definition 10 (Moments and Moment Generating Function, MGF). Given a convolution (with coefficients) \mathbf{f} , its i^{th} moment is defined as

$$M_i(\mathbf{f}) = \sum_{p=-\infty}^{+\infty} f_p p^i \quad (\text{B.13})$$

Furthermore, we define the moment generating function of the coefficients \mathbf{f} as

$$M_{\mathbf{f}}(\lambda) = \sum_{p=-\infty}^{+\infty} f_p e^{\lambda p} \quad (\text{B.14})$$

The name “moment generating” comes from the fact that

$$M_i(\mathbf{f}) = \left. \frac{d^i M_{\mathbf{f}}(\lambda)}{d\lambda^i} \right|_{\lambda=0} \quad (\text{B.15})$$

Remark: Since moment generating function (MGF) could be interpreted as a real-valued discrete-time Fourier transform (DTFT), the properties of MGF are very similar to the ones of DTFT. In particular, the convolution theorem also holds for MGF, i.e. $M_{\mathbf{f} * \mathbf{g}}(\lambda) = M_{\mathbf{f}}(\lambda) M_{\mathbf{g}}(\lambda)$. If two convolutions \mathbf{a} and $\bar{\mathbf{a}}$ are inverse to each other, we have $M_{\mathbf{a}}(\lambda) M_{\bar{\mathbf{a}}}(\lambda) = 1$.

Now we are ready to prove Theorem 3 using Theorem 8 and Definitions 9 and 10.

Proof. Let $\mathbf{f}^{(\ell)} = \bar{\mathbf{a}}^{(\ell)} * \mathbf{w}^{(\ell)}$, we have

$$\begin{aligned} \mathbf{y}^{(\ell)} &= \delta * \mathbf{y}^{(\ell)} = (\bar{\mathbf{a}}^{(\ell)} * \mathbf{a}^{(\ell)}) * \mathbf{y}^{(\ell)} = \bar{\mathbf{a}}^{(\ell)} * (\mathbf{a}^{(\ell)} * \mathbf{y}^{(\ell)}) \\ &= \bar{\mathbf{a}}^{(\ell)} * (\mathbf{w}^{(\ell)} * \mathbf{y}^{(\ell-1)}) = (\bar{\mathbf{a}}^{(\ell)} * \mathbf{w}^{(\ell)}) * \mathbf{y}^{(\ell-1)} = \mathbf{f}^{(\ell)} * \mathbf{y}^{(\ell-1)} \end{aligned} \quad (\text{B.16})$$

where each $\mathbf{f}^{(\ell)}$ has infinite number of coefficients. We denote the MGF of $\mathbf{f}^{(\ell)}$ as $M_{\mathbf{f}^{(\ell)}}$, and its first and second moments as $M_1(\mathbf{f}^{(\ell)})$ and $M_2(\mathbf{f}^{(\ell)})$. With the moments of $\mathbf{f}^{(\ell)}$, we can rewrite Equation B.1 in Theorem 8 as

$$r^2(\text{ERF}) = \sum_{\ell=1}^L \left[M_2(\mathbf{f}^{(\ell)}) - \left(M_1(\mathbf{f}^{(\ell)}) \right)^2 \right] \quad (\text{B.17})$$

The remaining part is to compute $M_{\mathbf{f}^{(\ell)}}$ for each $\mathbf{f}^{(\ell)}$, with which $M_1(\mathbf{f}^{(\ell)})$ and $M_2(\mathbf{f}^{(\ell)})$ are generated. Notice that $\mathbf{f}^{(\ell)} = \bar{\mathbf{a}}^{(\ell)} * \mathbf{w}^{(\ell)}$ is a convolution between $\bar{\mathbf{a}}^{(\ell)}$ and $\mathbf{w}^{(\ell)}$, we have

$$M_{\mathbf{f}^{(\ell)}}(\lambda) = M_{\bar{\mathbf{a}}^{(\ell)}}(\lambda) M_{\mathbf{w}^{(\ell)}}(\lambda) = \frac{M_{\mathbf{w}^{(\ell)}}(\lambda)}{M_{\mathbf{a}^{(\ell)}}(\lambda)} \quad (\text{B.18})$$

$$= \frac{1}{1 - a^{(\ell)} e^{\lambda}} \sum_{p=0}^{K^{(\ell)}-1} \frac{1 - a^{(\ell)}}{K^{(\ell)}} e^{\lambda p d^{(\ell)}} \quad (\text{B.19})$$

where the first equation uses the property that $M_{\mathbf{a}^{(\ell)}}(\lambda) M_{\bar{\mathbf{a}}^{(\ell)}}(\lambda) = 1$ for any λ . The first moment $M_1(\mathbf{f}^{(\ell)})$ is therefore

$$M_1(\mathbf{f}^{(\ell)}) = \left. \frac{dM_{\mathbf{f}^{(\ell)}}(\lambda)}{d\lambda} \right|_{\lambda=0} \quad (\text{B.20})$$

$$= \left\{ \frac{a^{(\ell)}}{(1 - a^{(\ell)} \lambda)^2} \sum_{p=0}^{K^{(\ell)}-1} \frac{1 - a^{(\ell)}}{K^{(\ell)}} e^{\lambda p d^{(\ell)}} + \frac{1}{1 - a^{(\ell)} e^{\lambda}} \sum_{p=0}^{K^{(\ell)}-1} \frac{1 - a^{(\ell)}}{K^{(\ell)}} p d^{(\ell)} e^{\lambda p d^{(\ell)}} \right\}_{\lambda=0} \quad (\text{B.21})$$

$$= \frac{a^{(\ell)}}{1 - a^{(\ell)}} + \frac{d^{(\ell)}}{K^{(\ell)}} \left(\sum_{p=0}^{K^{(\ell)}-1} p \right) \quad (\text{B.22})$$

$$= \frac{a^{(\ell)}}{1 - a^{(\ell)}} + \frac{d^{(\ell)} (K^{(\ell)} - 1)}{2} \quad (\text{B.23})$$

where the last equation makes use of Equation B.7a. Similarly, the second moment $M_2(\mathbf{f}^{(\ell)})$ is

$$M_2(\mathbf{f}^{(\ell)}) = \left. \frac{d^2 M_{\mathbf{f}^{(\ell)}}(\lambda)}{d\lambda^2} \right|_{\lambda=0} \quad (\text{B.24})$$

$$= \left\{ \frac{a^{(\ell)2}}{(1-a^{(\ell)})^3} \sum_{p=0}^{K^{(\ell)}} \frac{1-a^{(\ell)}}{K^{(\ell)}} e^{\lambda p d^{(\ell)}} + \frac{2a^{(\ell)}}{(1-a^{(\ell)}e^\lambda)^2} \sum_{p=0}^{K^{(\ell)}-1} \frac{1-a^{(\ell)}}{K^{(\ell)}} p d^{(\ell)} e^{\lambda p d^{(\ell)}} \right. \\ \left. + \frac{1}{1-a^{(\ell)}e^\lambda} \sum_{p=0}^{K^{(\ell)}-1} \frac{1-a^{(\ell)}}{K^{(\ell)}} (p d^{(\ell)})^2 e^{\lambda p d^{(\ell)}} \right\}_{\lambda=0} \quad (\text{B.25})$$

$$= \left(\frac{a^{(\ell)}}{1-a^{(\ell)}} \right)^2 + \frac{2a^{(\ell)}}{1-a^{(\ell)}} \frac{d^{(\ell)}}{K^{(\ell)}} \left(\sum_{p=0}^{K^{(\ell)}-1} p \right) + \frac{d^{(\ell)2}}{K^{(\ell)}} \left(\sum_{p=0}^{K^{(\ell)}-1} p^2 \right) \quad (\text{B.26})$$

$$= \left(\frac{a^{(\ell)}}{1-a^{(\ell)}} \right)^2 + \frac{2a^{(\ell)}}{1-a^{(\ell)}} \frac{d^{(\ell)} (K^{(\ell)} - 1)}{2} + \frac{d^{(\ell)2} (K^{(\ell)} - 1) (2K^{(\ell)} - 1)}{6} \quad (\text{B.27})$$

Plugging Equation B.23 and Equation B.27 into Equation B.17, we have

$$r^2(\text{ERF}) = \sum_{\ell=1}^L \left[\frac{d^{(\ell)2} (K^{(\ell)} - 1)^2}{12} + \frac{a^{(\ell)}}{(1-a^{(\ell)})^2} \right] \quad (\text{B.28})$$

Taking square root on both sides completes the proof. \square

ERF Analysis of ARMA Networks. If we assume all layers are identical, i.e. $K^{(\ell)} = K, d^{(\ell)} = d, a^{(\ell)} = a$ for $1 \leq \ell \leq L$, we can simplify Equation B.28 as

$$r(\text{ERF}) = \sqrt{L} \cdot \sqrt{\frac{d^2(K^2 - 1)}{12} + \frac{a}{(1-a)^2}} = O\left(\sqrt{L} \max\left(dK, \frac{\sqrt{a}}{1-a}\right)\right) \quad (\text{B.29})$$

The ERF radius is dominated by the AR coefficient when $a \lesssim 1$ regardless of kernel size K and dilation d . The radius still grows sub-linearly with the number of layers L in the linear network.

C Computation of ARMA Layers

In the section, we first derive the backpropagation rules in Theorem 4. We then show how to efficiently compute both forward and backward passes in ARMA layer using *Fast Fourier Transform*.

C.1 Backpropagation in ARMA models

In this part, we will prove a general theorem for backpropagation in ARMA models. To keep the notations simple, we derive the backpropagation equations for ARMA models with one dimension input/output and one channel. However, the techniques in the proof can be trivially extended to general ARMA models with high-dimensional input/output with multiple channels.

Theorem 11 (Backpropagation in an ARMA model). *Consider an ARMA model $\mathbf{a} * \mathbf{y} = \mathbf{w} * \mathbf{x}$, where \mathbf{a} and \mathbf{w} are the sequences of moving-average and autoregressive coefficients respectively, the gradients $\{\partial \mathcal{L} / \partial \mathbf{x}, \partial \mathcal{L} / \partial \mathbf{w}, \partial \mathcal{L} / \partial \mathbf{a}\}$ can be computed from $\partial \mathcal{L} / \partial \mathbf{y}$ with the following equations:*

$$\mathbf{a}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{w}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \quad (\text{C.1a})$$

$$-\mathbf{a}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \mathbf{y}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \quad (\text{C.1b})$$

$$\mathbf{a}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{x}^\top * \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \quad (\text{C.1c})$$

where $\mathbf{a}^\top, \mathbf{w}^\top$ and \mathbf{y}^\top denote the reversed sequences of \mathbf{a}, \mathbf{w} and \mathbf{y} respectively.

Notice that Theorem 4 is special case of Theorem 11: the first equation in Equation 5 is proved by Equation C.1b, and the second equation is proved by Equation C.1a.

We provide two different proofs of Theorem 11. **(1)** The analysis in our first proof is based on real numbers, and applicable to arbitrary types of convolution. **(2)** If the convolution is *circular* (as in the implementation of this paper), we provide a simpler proof using Fourier transform (therefore complex numbers). The second proof also suggests an FFT-based algorithm to compute the backpropagation in Equation 5 efficiently.

C.1.1 Proof in Real Numbers \mathbb{R}

Before we prove the theorem, we first prove a useful lemma on the inverse of transposed convolution.

Lemma 12 (Inverse of transposed convolution). *Given a convolution (with coefficients) \mathbf{a} , the operations of inversion and transposition are exchangeable,*

$$\overline{\mathbf{a}^\top} = \overline{\mathbf{a}}^\top \quad (\text{C.2})$$

that is, the inverse transposed convolution is equal to the transposed inverse convolution.

Proof. The lemma is an immediate result of the definitions of inverse and transposed convolutions.

$$\sum_{p=-\infty}^{+\infty} a_p^\top \overline{a_{i-p}^\top} = \sum_{p=-\infty}^{+\infty} a_{-p} \overline{a_{p-i}} = \delta_{-i} = \delta_i \quad \forall i \quad (\text{C.3})$$

which shows the inverse of \mathbf{a}^\top , i.e. $\overline{\mathbf{a}^\top}$, is equal to $\overline{\mathbf{a}}^\top$. \square

Now we are ready to prove Theorem 11 at the beginning of this section.

Proof. To begin with, we write the ARMA model $\mathbf{a} * \mathbf{y} = \mathbf{w} * \mathbf{x}$ in its weighted-sum form:

$$\sum_{q=-\infty}^{+\infty} a_q y_{i-q} = \sum_{p=-\infty}^{+\infty} w_p x_{i-p}, \quad \forall i \quad (\text{C.4})$$

Taking derivative w.r.t. a_r on both sides, and since the right side is a constant w.r.t. a_r , we have

$$\frac{\partial \left(\sum_{q=-\infty}^{+\infty} a_q y_{i-q} \right)}{\partial a_r} = 0, \quad \forall i, r \quad (\text{C.5})$$

By *implicit function theorem*, the left hand side can be further expanded as

$$\frac{\partial \left(\sum_{q=-\infty}^{+\infty} a_q y_{i-q} \right)}{\partial a_r} = \sum_{q=-\infty}^{+\infty} \frac{\partial (a_q y_{i-q})}{\partial a_r} \quad (\text{C.6})$$

$$= \sum_{q \neq r} a_q \frac{\partial y_{i-q}}{\partial a_r} + \left(y_{i-r} + a_r \frac{\partial y_{i-r}}{\partial a_r} \right) \quad (\text{C.7})$$

$$= \sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial a_r} + y_{i-r} = 0, \quad \forall i, r \quad (\text{C.8})$$

Rearranging the equation above, we have

$$- \sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial a_r} = y_{i-r}, \quad \forall i, r \quad (\text{C.9a})$$

Repeating the procedure twice for the derivatives w.r.t. w_r and x_r , we have two similar equations:

$$\sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial w_r} = x_{i-r}, \quad \forall i, r \quad (\text{C.9b})$$

$$\sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial x_r} = a_{i-r}, \quad \forall i, r \quad (\text{C.9c})$$

Since Equation C.9a, Equation C.9b and Equation C.9c take the same form, we only precede with Equation C.9b and obtain $\partial\mathcal{L}/\partial\mathbf{w}$. The other two can be derived using the same arguments.

Notice that Equation C.9b can be rewritten as

$$\sum_{q=-\infty}^{+\infty} a_{i-q} \frac{\partial y_q}{\partial w_r} = x_{i-r}, \forall i, r \quad (\text{C.10})$$

by changing variable q to $i - q$. Since Equation C.10 holds for any i , we further introduce a new index l and change i to $i - l$ on both hand sides:

$$\sum_{q=-\infty}^{+\infty} a_{i-q-l} \frac{\partial y_q}{\partial w_r} = x_{i-r-l}, \forall i, r, l \quad (\text{C.11})$$

Now we convolve both hand sides with $\bar{\mathbf{a}}$, the inverse of \mathbf{a} . Then for all i and r , we have

$$\sum_{l=-\infty}^{+\infty} \bar{a}_l \left(\sum_{q=-\infty}^{+\infty} a_{i-q-l} \frac{\partial y_q}{\partial w_r} \right) = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \quad (\text{C.12})$$

$$\sum_{q=-\infty}^{+\infty} \left(\sum_{l=-\infty}^{+\infty} \bar{a}_l a_{i-q-l} \right) \frac{\partial y_q}{\partial w_r} = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \quad (\text{C.13})$$

$$\frac{\partial y_i}{\partial w_r} = \sum_{q=-\infty}^{+\infty} \delta_{i-q} \frac{\partial y_q}{\partial w_r} = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \quad (\text{C.14})$$

Subsequently, we apply the chain rule to obtain $\partial\mathcal{L}/\partial w_r$

$$\frac{\partial\mathcal{L}}{\partial w_r} = \sum_{i=-\infty}^{+\infty} \frac{\partial y_i}{\partial w_r} \frac{\partial\mathcal{L}}{\partial y_i} = \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i}, \forall r \quad (\text{C.15})$$

Finally, we convolve both hand sides with \mathbf{a}^\top , the transpose of \mathbf{a} , to obtain the ARMA form of backpropagation rule.

$$\sum_{r=-\infty}^{+\infty} a_{j-r}^\top \frac{\partial\mathcal{L}}{\partial w_r} = \sum_{r=-\infty}^{+\infty} a_{j-r}^\top \left(\sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i} \right) \quad (\text{C.16})$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i} \quad (\text{C.17})$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{l-r} x_{i-l} \frac{\partial\mathcal{L}}{\partial y_i} \quad (\text{C.18})$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{r-l}^\top x_{l-i} \frac{\partial\mathcal{L}}{\partial y_i} \quad (\text{C.19})$$

$$= \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \left(\sum_{r=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{r-l}^\top \right) x_{l-i} \frac{\partial\mathcal{L}}{\partial y_i} \quad (\text{C.20})$$

$$= \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \delta_{j-l} x_{l-i} \frac{\partial\mathcal{L}}{\partial y_i} \quad (\text{C.21})$$

$$= \sum_{i=-\infty}^{+\infty} x_{j-i}^\top \frac{\partial\mathcal{L}}{\partial y_i}, \forall j \quad (\text{C.22})$$

where the second last equality uses Lemma 12. Therefore, we prove $\mathbf{a}^\top * \partial\mathcal{L}/\partial\mathbf{w} = \mathbf{x}^\top * \partial\mathcal{L}/\partial\mathbf{y}$, i.e. Equation C.1c in the theorem. Equation C.1b and Equation C.1a can be proved similarly. \square

C.1.2 Proof in Complex Numbers \mathbb{C}

In this part, we provide an alternative proof of Theorem 11 using Fourier transform.

Proof. If both convolutions in $\mathbf{a} * \mathbf{y} = \mathbf{w} * \mathbf{x}$ are circular with period N , the celebrated *convolution theorem* relates the discrete Fourier transform of \mathbf{a} , \mathbf{y} , \mathbf{w} and \mathbf{x} with

$$A_l Y_l = W_l X_l \quad \begin{cases} A_l = \sum_{n=0}^{N-1} a_n \omega_N^{nl}, & Y_l = \sum_{n=0}^{N-1} y_n \omega_N^{nl} \\ W_l = \sum_{n=0}^{N-1} w_n \omega_N^{nl}, & X_l = \sum_{n=0}^{N-1} x_n \omega_N^{nl} \end{cases} \quad (\text{C.23})$$

where $\omega_N = \exp(-j2\pi/N)$ is the N -th root of unity. For brevity, we only prove the most difficult equation $-\mathbf{a}^\top * \partial \mathcal{L} / \partial \mathbf{a} = \mathbf{y}^\top * \partial \mathcal{L} / \partial \mathbf{y}$ (Equation C.1b) here, and the proofs for the other two equations can be obtained with minor modification.

Taking derivative w.r.t. A_k on both hand sides, we have

$$\begin{cases} A_l \frac{\partial Y_l}{\partial A_k} = 0, & l \neq k \\ A_l \frac{\partial Y_l}{\partial A_k} + Y_k = 0, & l = k \end{cases} \quad (\text{C.24})$$

Since $A_l \neq 0, \forall l$, the equation can be simplified as

$$\frac{\partial Y_l}{\partial A_k} = \begin{cases} 0, & l \neq k \\ -\frac{Y_k}{A_k}, & l = k \end{cases} \quad (\text{C.25})$$

Then we apply chain rule to obtain the gradient of A_k , which yields

$$\frac{\partial \mathcal{L}}{\partial A_k} = \sum_{l=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_l} \frac{\partial Y_l}{\partial A_k} = -\frac{Y_k}{A_k} \frac{\partial \mathcal{L}}{\partial Y_k} \quad (\text{C.26})$$

Again, since $A_k \neq 0, \forall k$, we can simplify the equation as

$$A_k \frac{\partial \mathcal{L}}{\partial A_k} = -Y_k \frac{\partial \mathcal{L}}{\partial Y_k} \quad (\text{C.27})$$

(Notice that the equation above suggests an efficient algorithm to evaluate the equation using FFT.) To precede, we apply the chain rule one more time to obtain the derivatives w.r.t. a_n and y_n .

$$\frac{\partial \mathcal{L}}{\partial a_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial A_k} \frac{\partial A_k}{\partial a_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial A_k} \omega_N^{kn} \quad (\text{C.28a})$$

$$\frac{\partial \mathcal{L}}{\partial y_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_k} \frac{\partial Y_k}{\partial y_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_k} \omega_N^{kn} \quad (\text{C.28b})$$

With the equations above, the convolution between \mathbf{a}^\top and $\partial \mathcal{L} / \partial \mathbf{a}$ can be rewritten as

$$\sum_{n=0}^{N-1} a_{i-n}^\top \frac{\partial \mathcal{L}}{\partial a_n} = \sum_{n=0}^{N-1} a_{n-i} \frac{\partial \mathcal{L}}{\partial a_{n-i}} \quad (\text{C.29})$$

$$= \sum_{n=0}^{N-1} a_{n-i} \left(\sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial A_k} \omega_N^{kn} \right) \quad (\text{C.30})$$

$$= \sum_{k=0}^{N-1} \left(\sum_{n=0}^{N-1} a_{n-i} \omega_N^{k(n-i)} \right) \frac{\partial \mathcal{L}}{\partial A_k} \omega_N^{ki} \quad (\text{C.31})$$

$$= \sum_{k=0}^{N-1} A_k \frac{\partial \mathcal{L}}{\partial A_k} \omega_N^{ki} \quad (\text{C.32})$$

With identical arguments, we can rewrite the convolution between \mathbf{y}^\top and $\partial\mathcal{L}/\partial\mathbf{y}$ as

$$\sum_{n=0}^{N-1} y_{i-n}^\top \frac{\partial\mathcal{L}}{\partial y_n} = \sum_{k=0}^{N-1} Y_k \frac{\partial\mathcal{L}}{\partial Y_k} \omega_N^{ki} \quad (\text{C.33})$$

Recall the relation in Equation C.27, we have

$$-\sum_{n=0}^{N-1} a_{i-n}^\top \frac{\partial\mathcal{L}}{\partial a_n} = \sum_{n=0}^{N-1} y_{i-n}^\top \frac{\partial\mathcal{L}}{\partial y_n} \quad (\text{C.34})$$

i.e. $-\mathbf{a}^\top * \partial\mathcal{L}/\partial\mathbf{a} = \mathbf{y}^\top * \partial\mathcal{L}/\partial\mathbf{y}$, which completes the proof. \square

C.2 Efficient Computation using Fast Fourier Transform

The key to speeding up both forward and backward passes in ARMA layers is the *Discrete Fourier Transform* (DFT), along with the *Fast Fourier Transform* (FFT) algorithm.

Definition 13 (Discrete Fourier Transform, DFT). Given a third-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times C}$, we define its DFT of over the spatial coordinates as $\tilde{\mathcal{T}} \in \mathbb{C}^{I_1 \times I_2 \times C}$.

$$\tilde{\mathcal{T}}_{k_1, k_2, c} = \sum_{i_1=0}^{I_1-1} \sum_{i_2=0}^{I_2-1} \mathcal{T}_{i_1, i_2, c} \omega_{I_1}^{-i_1 k_1} \omega_{I_2}^{-i_2 k_2} \quad (\text{C.35})$$

where $\omega_I = \exp(2\pi/I)$ is the I^{th} root of unity. Given the transformed tensor $\tilde{\mathcal{T}} \in \mathbb{C}^{I_1 \times I_2 \times C}$, the original tensor \mathcal{T} can be recovered by inverse DFT (IDFT) as

$$\mathcal{T}_{i_1, i_2, c} = \frac{1}{I_1 I_2} \sum_{k_1=0}^{I_1-1} \sum_{k_2=0}^{I_2-1} \tilde{\mathcal{T}}_{k_1, k_2, c} \omega_{I_1}^{i_1 k_1} \omega_{I_2}^{i_2 k_2} \quad (\text{C.36})$$

The definition above can be extended to convolutional kernels \mathcal{A} by first zero-padding \mathcal{A} to be $\mathbb{R}^{I_1 \times I_2 \times C}$. With DFT, the autoregressive layer in Equation 4 can be computed as

$$\tilde{\mathcal{A}}_{k_1, k_2, t} \tilde{\mathcal{Y}}_{k_1, k_2, t} = \tilde{\mathcal{T}}_{k_1, k_2, t} \quad (\text{C.37})$$

where $\tilde{\mathcal{A}}, \tilde{\mathcal{T}}$ are computed from \mathcal{A}, \mathcal{T} with Equation C.35, and \mathcal{Y} is recovered from $\tilde{\mathcal{Y}}$ by Equation C.36. Similarly, the backpropagation in Equation 5 can be solved as

$$\frac{\partial\mathcal{L}}{\partial \tilde{\mathcal{A}}_{k_1, k_2, t}} = -\frac{\tilde{\mathcal{Y}}_{k_1, k_2, t}}{\tilde{\mathcal{A}}_{k_1, k_2, t}} \cdot \frac{\partial\mathcal{L}}{\partial \tilde{\mathcal{Y}}_{k_1, k_2, t}} \quad (\text{C.38a})$$

$$\frac{\partial\mathcal{L}}{\partial \tilde{\mathcal{A}}_{k_1, k_2, t}} = \frac{1}{\tilde{\mathcal{A}}_{k_1, k_2, t}} \cdot \frac{\partial\mathcal{L}}{\partial \tilde{\mathcal{Y}}_{k_1, k_2, t}} \quad (\text{C.38b})$$

If every DFT is evaluated using FFT, the computational complexity of either forward or backward pass reduces to $O(\log(\max(I_1, I_2))I_1 I_2 T)$, compared to $O((I_1^2 + I_2^2)I_1 I_2 T)$ using Gaussian elimination.

D Stability of ARMA Layers

In this section, we will prove the main Theorem 6 in section 5. The section is organized in three subsections: (1) In subsection D.1, we formally define the concept of *BIBO stability*, and prove a lemma that relates the stability of a complicated model to the ones of its submodules; (2) In subsection D.2, we repeatedly apply the lemma and deduce the stability of an ARMA layer to from the one of length-3 filters; (3) Lastly in subsection D.3, we prove a theorem on the stability of a length-3 filter.

D.1 Algebra of BIBO stability

To analyze the stability of an ARMA model, we adopt the traditional notion of *Bounded-Input Bounded-Output* (BIBO) stability [28] that characterizes stability of linear systems.

Definition 14 (BIBO stability). An input \mathbf{x} (or an output \mathbf{y}) is bounded if $|x_i| < B_1, \forall i \in \mathbb{Z}$ for some $B_1 > 0$ (or $|y_i| < B_2, \forall i \in \mathbb{Z}$ for some $B_2 > 0$). A model is BIBO stable if the output \mathbf{y} is bounded given any bounded input \mathbf{x} , that is

$$\forall \mathbf{x}, (\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_2 > 0, |y_i| < B_2, \forall i \in \mathbb{Z}) \quad (\text{D.1})$$

The following lemma presents that the BIBO stability is preserved under simple algebraic operations of *cascade*, *addition* and *concatenation*. This lemma allows us to reduce the stability analysis of a complex model into its simpler submodules.

Lemma 15 (Preserved BIBO Stability). BIBO stability is preserved under the operations of *cascade*, *addition* and *concatenation*. Suppose f and g are two BIBO stable models, and consider three compound models: **(1)** $h_1 = g \circ f$ is a cascaded model $\mathbf{y} = h_1(\mathbf{x}) = g(f(\mathbf{x}))$, **(2)** $h_2 = f + g$ is a parallel model $\mathbf{y} = h_2(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$, **(3)** $h_3 = f \otimes g$ is a concatenated model $\mathbf{y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}] = h_3([\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]) = [f(\mathbf{x}^{(1)}), g(\mathbf{x}^{(2)})]$, h_1 , h_2 and h_3 are all BIBO stable.

Proof. (1) Cascaded model $h_1 = g \circ f$: $\mathbf{y} = h_1(\mathbf{x}) = g(f(\mathbf{x}))$. Let $\mathbf{t} = f(\mathbf{x})$ denote the intermediate result returned by the model f . Since f is BIBO stable, we have

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_0 > 0, |t_i| < B_0, \forall i \in \mathbb{Z}) \quad (\text{D.2a})$$

Similarly, since g is BIBO stable, we further have

$$(\exists B_0 > 0, |t_i| < B_0, \forall i \in \mathbb{Z}) \implies (\exists B_2 > 0, |y_i| < B_2, \forall i \in \mathbb{Z}) \quad (\text{D.2b})$$

Combining both Equation D.2a and Equation D.2b, we achieve

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_2 > 0, |y_i| < B_2, \forall i \in \mathbb{Z}) \quad (\text{D.3})$$

which is the definition of BIBO stability for model h_1 .

(2) Parallel model $h_2 = f + g$: $\mathbf{y} = h_2(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$. Let $\mathbf{u} = f(\mathbf{x})$ and $\mathbf{v} = g(\mathbf{x})$ be the outputs of f and g . Since both f and g are BIBO stable, we have the following two relations:

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_{21} > 0, |u_i| < B_{21}, \forall i \in \mathbb{Z}) \quad (\text{D.4a})$$

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_{22} > 0, |v_i| < B_{22}, \forall i \in \mathbb{Z}) \quad (\text{D.4b})$$

Combining both Equation D.4a and Equation D.4b, we have

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (|y_i| < B_2 = B_{21} + B_{22}, \forall i \in \mathbb{Z}) \quad (\text{D.5})$$

We achieve the definition BIBO stability for model h_2 .

(3) Concatenated model $\mathbf{y} = f \otimes g$: $\mathbf{y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}] = h([\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]) = [f(\mathbf{x}^{(1)}), g(\mathbf{x}^{(2)})]$. Since f and g are both BIBO stable, we have the following relations:

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_{21} > 0, |y_i^{(1)}| < B_{21}, \forall i \in \mathbb{Z}) \quad (\text{D.6a})$$

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (\exists B_{22} > 0, |y_i^{(2)}| < B_{22}, \forall i \in \mathbb{Z}) \quad (\text{D.6b})$$

Again, combining both equations we have

$$(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \implies (|y_i^{(2)}| < B_2 = \max(B_{21}, B_{22}), \forall i \in \mathbb{Z}) \quad (\text{D.7})$$

And we achieve the BIBO stability for model h_3 . \square

D.2 Reduction of an ARMA layer

In what follows, we repeatedly use Lemma 15 to decompose an ARMA layer into simpler submodules until the stability analysis for the submodule is tractable.

From ARMA model to AR model. In section 4, we show that an ARMA layer can be decomposed into a *cascade* of a *traditional convolutional layer* and an *autoregressive layer* in Equation 4. Since the traditional convolutional layer is always BIBO stable, it is sufficient to guarantee the stability of the autoregressive layer:

$$\mathcal{A}_{:,t} * \mathcal{Y}_{:,t} = \mathcal{T}_{:,t}, \forall t \quad (\text{D.8})$$

From multiple channels to a single channel. Note that the autoregressive layer in Equation D.8 is a *concatenation* of T channels of ARMA models, therefore it is sufficient to guarantee the stability of each ARMA model. For simplicity, we drop the subscript t and denote $\mathcal{A}, \mathcal{Y}, \mathcal{T}$ as $\mathbf{A}, \mathbf{Y}, \mathbf{T}$. Our goal now reduces to finding a sufficient condition for the stability of

$$\mathbf{A} * \mathbf{Y} = \mathbf{T} \iff \sum_{p_1, p_2} A_{p_1, p_2} Y_{i_1 - p_1, i_2 - p_2} = T_{i_1, i_2}, \forall i_1, i_2 \quad (\text{D.9})$$

where $\mathbf{A} \in \mathbb{R}^{K_a \times K_a}$ and $\mathbf{T}, \mathbf{Y} \in \mathbb{R}^{I_1 \times I_2}$.

From separable 2D-filter to two 1D-filters. In a *separable ARMA layer* (Equation 6), each filter \mathbf{A} in Equation D.9 is separable, i.e. $\mathbf{A} = \mathbf{f} \otimes \mathbf{g}$ is outer product of two 1D-filters $\mathbf{f} \in \mathbb{R}^{I_1}, \mathbf{g} \in \mathbb{R}^{I_2}$:

$$A_{p_1, p_2} = f_{p_1} g_{p_2}, \forall p_1, p_2 \quad (\text{D.10})$$

Given the factorization, the model in Equation D.9 can be written as a cascade of two submodules:

$$\sum_{p_1} f_{p_1} S_{i_1 - p_1, i_2} = T_{i_1, i_2}, \forall i_2 \quad (\text{D.11a})$$

$$\sum_{p_2} g_{p_2} Y_{i_1, i_2 - p_2} = S_{i_1, i_2}, \forall i_1 \quad (\text{D.11b})$$

where $\mathbf{S} \in \mathbb{R}^{I_1 \times I_2}$ is an intermediate result. Notice that Equation D.11a is a concatenation of I_2 submodules, each of which operates on a column of \mathbf{T} . Similarly, Equation D.11b can be decomposed into a concatenation of I_1 submodules, and each submodule operates on a row of \mathbf{S} . According to Lemma 15, it is sufficient to guarantee the stability of \mathbf{f} and \mathbf{g} individually. For simplicity, we denote both \mathbf{f} and \mathbf{g} as \mathbf{a} , and rewrite each submodule in Equation D.11a or Equation D.11b as

$$\mathbf{a} * \mathbf{y} = \mathbf{x} \iff \sum_p a_p y_{i-p} = x_i, \forall i \quad (\text{D.12})$$

From general 1D-filter to composition of length-3 filters. By the *fundamental theorem of algebra*, any one-dimensional filter can be decomposed as a composition of shorter filters [28]. Specifically, suppose $\mathbf{a} \in \mathbb{R}^K$ is a filter of length- K , it can be factorized into a composition of $Q = (K-1)/2$ length-3 filters such that

$$\mathbf{a} = \mathbf{a}^{(1)} * \mathbf{a}^{(2)} \dots * \mathbf{a}^{(Q)} \quad (\text{D.13})$$

where each filter $\mathbf{a}^{(q)} \in \mathbb{R}^3$ has three coefficients. By the decomposition, the model in Equation D.12 is a cascade of Q submodules

$$\mathbf{a}^{(1)} * \left(\mathbf{a}^{(2)} * \dots \left(\mathbf{a}^{(Q)} * \mathbf{y} \right) \right) = \mathbf{x} \quad (\text{D.14})$$

Therefore, we only need to guarantee the stability for each $\mathbf{a}^{(q)}$ individually. In the next subsection, we will further drop the superscript q and assume \mathbf{a} itself is a length-3 filter.

D.3 Stability of a length-3 1D-filter

Without loss of generality, we assume the filter \mathbf{a} is centered at 0 with $a_0 = 1$ (otherwise we can rescale the moving-average coefficients). The model at consideration can be written as

$$a_1 y_{i-1} + y_i + a_{-1} y_{i+1} = x_i \quad (\text{D.15})$$

The stability analysis of this model follows the standard approach of Z-transform [28]. To begin with, we review the concepts of Z-transform, *Region of Convergence* (ROC) and their relationships to BIBO stability of a linear model.

Definition 16 (Z-transform and ROC). Given a one-dimensional sequence \mathbf{h} , the Z-transform maps the sequence to a complex function on the complex plane \mathbb{C}

$$H(z) = \sum_{i=-\infty}^{+\infty} h_i z^{-i} \quad (\text{D.16})$$

Notice that the infinite series does not necessarily converge for any $z \in \mathbb{C}$, and the transformation exists only if the summation is convergent. The region in the complex plane that the Z-transform exists is known as the ROC for the sequence \mathbf{h} .

Lemma 17 (ROC and BIBO stability). Consider a linear model $\mathbf{y} = \mathbf{h} * \mathbf{x}$, and let H denote the Z-transform of \mathbf{h} , then a necessary and sufficient condition for the model being BIBO stable is that the unit circle belongs to the ROC, i.e. the infinite series

$$H(e^{j\omega}) = \sum_{i=-\infty}^{+\infty} h_i e^{-j\omega i} \quad (\text{D.17})$$

converges for any frequency $\omega \in \mathbb{R}$, i.e. discrete-time Fourier transform (DTFT) exists for \mathbf{h} .

Lemma 18 (ROC of length-3 AR model). Consider an length-3 AR model $\mathbf{a} * \mathbf{y} = \mathbf{x}$, i.e. $a_{-1}y_{i-1} + y_i + a_1y_{i+1} = x_i$, the Z-transform of \mathbf{a} is a length-3 complex polynomial $A(z) = a_{-1}z + 1 + a_1z^{-1}$ with two zeros z_1 and z_2 . Then the Z-transform of its inverse convolution $\bar{\mathbf{a}}$ is

$$\bar{A}(z) = \frac{1}{A(z)} = \frac{z}{a_{-1}z^2 + z + a_1} \quad (\text{D.18})$$

with the corresponding ROC $|z_1| < z < |z_2|$ as a ring. Since the model can be written as $\mathbf{y} = \bar{\mathbf{a}} * \mathbf{x}$, it is BIBO stable if $|z_1| < 1 < |z_2|$ according to Lemma 17.

With the lemmas above, we are ready to prove Theorem 6.

Proof. Since the coefficients in \mathbf{a} are real numbers, the zeros of $F(z) = zA(z) = a_{-1}z^2 + z + a_1$ are conjugate to each other: **(1)** Both zeros lie on the real axis, i.e. z_1 and z_2 are real numbers; and **(2)** z_1 and z_2 are complex conjugate to each other, i.e. $z_1^* = z_2$.

Notice that **(2)** also implies $|z_1| = |z_2|$. However, Lemma 18 shows that $|z_1| < 1 < |z_2|$ is required for BIBO stability, and therefore the second distribution is not feasible.

If both zeros are real, the inequality $|z_1| < 1 < |z_2|$ is equivalent to $F(1) \cdot F(-1) < 0$, i.e.

$$(a_{-1} + 1 + a_1)(a_{-1} - 1 + a_1) < 0 \quad (\text{D.19})$$

$$(a_{-1} + a_1)^2 - 1 < 0 \implies |a_{-1} + a_1| < 1 \quad (\text{D.20})$$

which completes the proof. \square

The constrain $|a_{-1} + a_1| < 1$ can be removed by re-parameterizing (a_{-1}, a_1) into (α, β) :

$$\begin{pmatrix} a_{-1} \\ a_1 \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} \begin{pmatrix} \alpha \\ \tanh(\beta) \end{pmatrix} \quad (\text{D.21})$$

where the learnable parameters (α, β) have no constrain. The transform in re-parameterization is illustrated in the following figure.

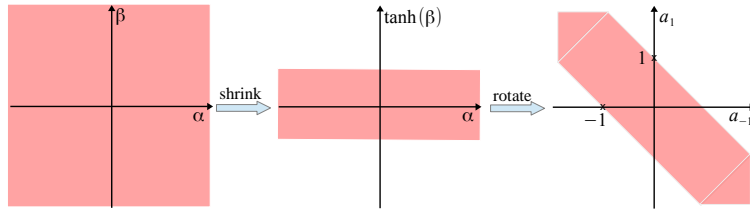


Figure 17: Visualization of the re-parameterization in Equation D.21.